Data Management in the Noisy Intermediate-Scale Quantum Era: Extended Version

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@liacs.leidenuniv.nl

Tim Littau
Delft University of Technology
t.m.littau@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 [7, 152]. 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 [122]. 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 [133].

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 [122, 169]. 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, 179], and error correction codes [57, 58, 67, 154]. 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, 100, 177, 179]. 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 [53, 54, 120, 140–143, 159, 173], data integration [59], index selection [70, 87], and transaction management [19, 69, 157]. 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, 71, 174, 180, 181] 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, 135, 177, 179]. 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, and report on preliminary experimental results (Sec. 4).

2 DATA IN THE QUANTUM ERA

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 [149]. 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 [122]. 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. 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.

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* [132], 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 [161]. 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 [122].

3. Quantum data can be entangled. Another difference between qubits and bits is *entanglement* [50], leading to *spooky action at a distance*, 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. A well-known entangled state is the *Bell's state*:

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

where knowledge of one qubit determines the other. See Appendix A of our online report [6] for more background.

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 paradigms 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. Our goal is to introduce data management researchers to quantum use cases, and by making a distinction based on the nature and role of data within cases. We primarily present future data management research for quantum computing. In Sec. 4, we provide concrete research directions in these paradigms.

I Classical simulation of quantum computing paradigm: classical data represents quantum states and operations. This paradigm presents great potential for new database challenges. It focuses on using *classical data to represent and simulate quantum data*. For example, Bell's state from Sec. 2 is represented by a classical vector. This paradigm is more accessible as it relies on classical computers, not quantum ones.

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 [177, 179].[1] Simulations are of paramount importance in the NISQ era [133]. 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, 84, 166, 184, 186]. 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 [177].

The potential for database research in this paradigm is immense. First, imagine innovative database systems specifically designed to manage classical data that supports simulation. Second, efficient caching strategies for expensive operations, such as repeated simulations of quantum circuits with varying error parameters, present another important research direction. The third research direction in this paradigm involves the representation of quantum states and operations. This is useful for quantum design questions such as optimizing the number of quantum gates or gate depth [90, 185] and compiling textbook quantum circuits to only include the low-level operations that real quantum devices support [148]. 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, 125] or other advanced representations [27, 86, 162, 186].

[1] In this work, by simulation we refer to *classical simulation*. Another related term is *quantum simulation* [30, 64], which pertains to simulating quantum mechanics on a quantum computer, a subject studied in physics.

II Joint Quantum-Classical Computing paradigm: classical preprocessing & postprocessing for quantum computing. This paradigm focuses on the situation involved in 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—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, 131]. This is followed by classical postprocessing on the bitstring (i.e. classical data) that the measurement returns [51]. 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 [67, 154], whose outcomes are then decoded to find out which error has most likely occurred [14, 109]. Third, efficient preprocessing is critical for quantum algorithms, which requires *loading* or *encoding* 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 [153] or to output a quantum state with amplitudes representing the classical data [40]. This is a challenging problem. For example, the Harrow–Hassidim–Lloyd (HHL) algorithm, a well-known quantum approach for solving a system of linear equations [73], achieves exponential speedup over classical methods. 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 is an open research question.

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] which rotates its resources among multiple clients, various types of quantum machine learning [32], and Quantum Random Access Memory [65], 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.

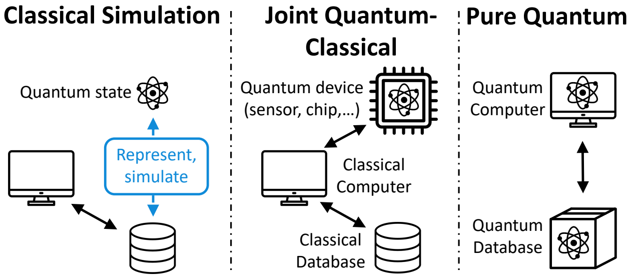


Figure 1: Landscape: data management for quantum computing

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, 55, 62, 77, 113, 146] and efficient use of different quantum hardware types each of which has speed-decoherence trade-offs [47]. 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 Paradigm 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). Then, a simulation is a classical computation that computes that output distribution (so-called *strong* simulation), or samples from it (*weak* simulation) [164]. We primarily focus on strong simulation.

The major practical bottleneck of (strong) simulation is *scalability* [25, 100, 177, 179]. When simulating a quantum state of n qubits, the state is typically represented as a vector with a size of 2^n. The

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

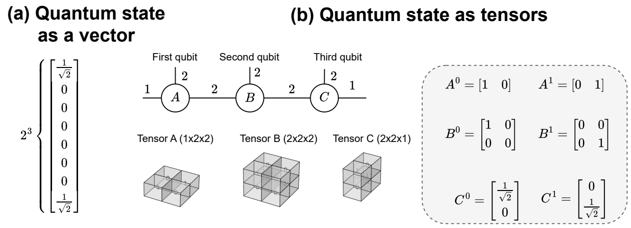

(a) Quantum state as a vector
(b) Quantum state as tensors
2^3
$\frac{1}{\sqrt{2}}$, 0, 0, 0, 0, 0, 0, $\frac{1}{\sqrt{2}}$
First qubit
Second qubit
Third qubit
1
2
A
B
C
Tensor A (1x2x2)
Tensor B (2x2x2)
Tensor C (2x2x1)
$A^0 = [1\ \ 0]$
$A^1 = [0\ \ 1]$
$B^0 = \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}$
$B^1 = \begin{bmatrix}0 & 0\\0 & 1\end{bmatrix}$
$C^0 = \begin{bmatrix}\frac{1}{\sqrt{2}}\\0\end{bmatrix}$
$C^1 = \begin{bmatrix}0\\\frac{1}{\sqrt{2}}\end{bmatrix}$

Figure 2: (a) The 3-qubit GHZ state $\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$ represented as a state vector of size $2^3 = 8$. (b) GHZ state as a tensor network. 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. See Appendices A and B1 of [6] for details.

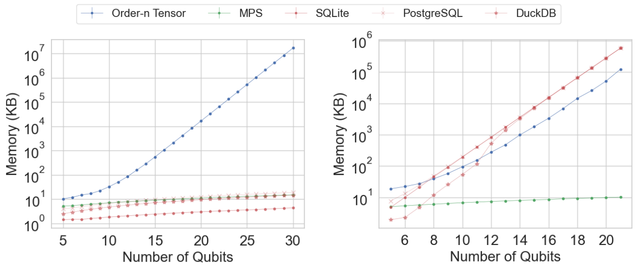
size of the state vector grows exponentially as n increases. For instance, an experiment demonstrating quantum supremacy required 2.25 petabytes for 48 qubits, reaching the memory limits of today's supercomputers [25]. To overcome the memory restriction, existing simulation tools have explored approximation [85, 111, 165], data compression [175], parallelization [79], and distributed computing [147]. 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 contraction is the summation of shared indices between tensors and is a generalization of matrix multiplication [18]. See Appendix A of [6] for more details. The first question we ask is as follows.



(a) GHZ State (sparse circuit)

(b) QFT (dense circuit)

Figure 3: 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. For sparse circuit (GHZ), RDBMS solutions demonstrate significantly lower memory usage than the order-n tensor baseline, with SQLite showing a slight advantage over the MPS baseline. For the dense circuit QFT, the memory usage of RDBMS solutions grows comparably to or exceeds the order-n tensor baseline, indicating the need for improvements in handling dense tensor computations.

> **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, 158] 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 [158] introduces the idea of supporting dense vectors using SQL UNION ALL and CASE statements. Despite these 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. Moreover, our preliminary results in Fig. 3 demonstrate that RDBMS-based simulations are efficient for quantum circuits involving sparse tensor computations. All experimental details, together with a space and time complexity analysis, can be found in Appendix B of [6].[5] RDBMS-based solutions are competitive with, and may even outperform, state-of-the-art representations like MPS in specific scenarios. Interestingly, this advantage does not extend to dense circuits such as quantum fourier transform (QFT) circuits, highlighting the need for further optimizations to enhance RDBMS systems for simulation workloads. In Appendix C of [6] we additionally discuss the use of NoSQL databases.

Q1 summary. Pushing simulation workloads to DBMSs leverages their efficiency, showing promise for sparse circuits, yet challenges in optimization and scalability remain. On the other hand, the connection to tensor computations suggests the following question.

[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 [66].

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

[5] Codes are available at https://github.com/InfiniData-Lab/Quantum

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, 89, 105, 126, 183]. These studies range from high-level representations to the runtime optimization of ML workflows. At the representation level, numerous studies represent data as tensors [93] and aim to integrate relational and tensor operations [81, 88, 137]. Meanwhile, system-building endeavors from the database community [21, 21, 61, 74, 103, 117, 123, 129, 138, 139] focus on performance efficiency and scalability, employing optimization 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, 83, 91, 136].

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, 139].

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

To 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 [43, 167], C++ [9, 167] or Julia [104]. There are also dedicated quantum programming languages and domain-specific compilation techniques. See the survey [75]. 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* [163]. Its graphical nature enables simplifications by locally merging connected vertices while preserving the original tensor, useful in various contexts including formal verification [46, 98, 163].

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 (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 by 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 [99, 151, 160]. 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 [43, 179]. 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 [179]. Quantum-specific issues, such as quantum noise (Sec. 2), 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. Particular to the quantum setting is the noisy character of the quantum computation, which is known to impact the efficiency of simulation [8, 78, 79, 128, 134]. 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, 92]. Having algorithms in place for testing equivalence may help to avoid redundant computations during simulation.

Q3 summary. Quantum-specific optimizations require a dedicated simulation optimizer to handle tensor contraction, state sparsity, noise levels, and task-specific strategies, while also eliminating redundant computations through equivalence checking.

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 [169], 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 [45]. The choice of data representation can significantly impact whether certain quantum operations (e.g., gates, measurement explained in Sec. 2 and Appendix A of our technical report [6]) are tractable [169].

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, 86, 162, 186]. 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) [168]. 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 [169]. 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 [96].

Q4 summary. Effective data representations for quantum simulations require a balance of 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 remain in this context when considering quantum simulation. We prioritize Q1–Q4, however, as first steps. A more extensive discussion can be found in Appendix D in [6].

4.3 More data management opportunities

Beyond simulation for quantum computing, quantum-related research is broad, e.g., error correction [67, 145, 150], quantum networks [44, 172]. 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 quantum computation on noisy quantum processors and is a crucial part of the roadmap to fault-tolerant quantum computation [154]. 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, 57, 58]. Intermittently, decoding is applied: errors are detected and corrected. Decoders [48, 56, 176] 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, 106, 110, 171, 178], 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 [42]. Another compelling direction is to explore how to build specialized data lakes [72, 118] or lakehouses [11, 12, 82] 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 [102, 112]. 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 [170]. An interesting new direction is *data-driven QEC*, which employs ML techniques to quantum error correction, such as RL [10, 119, 182], multilayer perceptrons (MLP) [37], convolutional neural networks [33], and GNN [97]. 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 [52, 60] and MLP [127], have shown potential in optimizing circuit design to enhance the efficiency of quantum circuits on real devices.

5 CONCLUSION

We are 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. We here identify the unique features of quantum data compared to classical data, and advocate 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] 2025. Data Management in the Noisy Intermediate-Scale Quantum Era: Extended Version. https://github.com/infiniData-Lab/Quantum/blob/main/technical_report.pdf

[7] 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.

[8] 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.

[9] Quantum AI. 2021. qsim and qsimh. https://quantumai.google/qsim/overview. Accessed: 2024-08-05.

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

[41] 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.

[42] 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

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

[44] 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.

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

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

[47] 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.

[48] 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]

[49] Wolfgang Dür, Guifre Vidal, and J Ignacio Cirac. 2000. Three qubits can be entangled in two inequivalent ways. *Physical Review A* 62, 6 (2000), 062314.

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

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

[52] 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.

[53] 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]

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

[55] 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).

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] 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

[61] 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.

[62] 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.

[63] 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.

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

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

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

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

[68] Daniel M Greenberger, Michael A Horne, and Anton Zeilinger. 1989. Going beyond Bell's theorem. In *Bell's theorem, quantum theory and conceptions of the universe*. Springer, 69–72.

[69] 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.

[70] 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.

[71] 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.

[72] 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.

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

[74] 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.

[75] 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.

[76] Paul Herringer and Robert Raussendorf. 2023. Classification of measurement-based quantum wire in stabilizer PEPS. *Quantum* 7 (June 2023), 1041. https://doi.org/10.22331/q-2023-06-12-1041

[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] 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)*.

[83] 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.

[84] 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.

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

[86] 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.

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

[88] 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.

[89] 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).

[90] 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.

[91] 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.

[92] 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.

[93] 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.

[94] 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

[95] 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.

[96] 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

[97] 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

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

[99] 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.

[100] 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.

[101] 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.

[102] 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.

[103] 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.

[104] 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.

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

[106] 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.

[107] 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

[108] Igor L. Markov and Yaoyun Shi. 2008. Simulating Quantum Computation by Contracting Tensor Networks. *SIAM J. Comput.* 38, 3 (2008), 963–981.

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

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

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

[112] 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

[113] 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.

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

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

[116] Avinash Mocherla, Lingling Lao, and Dan E Browne. 2023. Extending matchgate simulation methods to universal quantum circuits. *arXiv preprint arXiv:2302.02654* (2023).

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

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

[123] 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.

[124] Bryan O'Gorman. 2019. Parameterization of tensor network contraction. *arXiv preprint arXiv:1906.00013* (2019).

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

[126] 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.

[127] 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.

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

[129] 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.

[130] 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.

[131] 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.

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

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

[134] 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.

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

[136] 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.

[137] 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.

[138] 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.

[139] 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.

[140] 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.

[141] 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.

[142] 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.

[143] 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.

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

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

[146] 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]

[147] 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]

[148] 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.

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

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

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

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

[153] 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

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

[155] Dimitrios Thanos, Alejandro Villoria, Sebastiaan Brand, Arend-Jan Quist, Jingyi Mei, Tim Coopmans, and Alfons Laarman. 2024. Automated reasoning in quantum circuit compilation. In *International Symposium on Model Checking Software*. Springer, 106–134.

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

[157] 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.

[158] 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)*.

[159] 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).

[160] 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.

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

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

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

[164] 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.

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

[166] 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.

[167] 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.

[168] 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.

[169] 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

[170] 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

[171] 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.

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

[173] 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)*.

[174] 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.

[175] 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.

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

[177] 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]

[178] 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.

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

[180] 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]

[181] 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.

[182] 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

[183] 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).

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

[185] 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.

[186] 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.

Appendices

Appendix A BACKGROUND: DATA REPRESENTATION FOR SIMULATION

We here describe several existing 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 normalization constraint $\sum_{j=0}^{2^n-1} |v_j|^2 = 1$, where $|.|$ denotes the norm or modulus of a complex number [122]. See Fig. 4(a). The number v_j is called the amplitude corresponding to the index $j \in \{0, 1, \dots, 2^n - 1\}$. 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 $k \in \{0, 1, \dots, 2^n - 1\}$ (see Fig. 4(a1) for examples). Each additional qubit thus adds a single bit to the address, consequently doubling the size of the state vector.

The canonical input state to an n-qubit quantum circuit is 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 $O(N^2)$.

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 (note that every matrix is an order-2 tensor and every vector an order-1 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 labeled with the corresponding dimension, see Fig. 4(a2, b2).

Two edges with the same dimension can be connected; contracting this connection 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 $C_{ijklm} = \sum_{x=1}^{3} A_{ixj} \cdot B_{xklm}$. 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. 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 [130]. Formally, an MPS representing an n-qubit state is equivalently defined as a collection of $2n$ matrices $M_1^0, M_1^1, M_2^0, \dots, M_n^0, M_n^1$, such that the entry of the quantum state vector at binary address $x_1 x_2 \dots x_n \in \{0, 1\}^n$ is found as the (single entry of the 1×1) matrix product $\prod_{j=1}^{n} M_j^{x_j}$. For this product to be a 1×1 matrix, the input dimension of M_1^0 and M_1^1 needs to be 1 (i.e. they are lying vectors), and similarly the output dimension of M_n^0 and M_n^1 is 1 (i.e. they are standing vectors).

For example, the following MPS represents the 3-qubit GHZ-state (see its state vector in Fig. 4(a1)):

$$M_1^0 = \begin{pmatrix} 1 & 0 \end{pmatrix} \quad M_1^1 = \begin{pmatrix} 0 & 1 \end{pmatrix}$$

$$M_2^0 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad M_2^1 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

$$M_3^0 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} \quad M_3^1 = \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

For example, the GHZ-state's entry $\frac{1}{\sqrt{2}}$ at the top of the vector, i.e. binary index 000, is found as the matrix product

$$M_1^0 \cdot M_2^0 \cdot M_3^0 = \begin{pmatrix} 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \end{pmatrix}$$

while the entry with value 0 at binary index 101 equals

$$M_1^1 \cdot M_2^0 \cdot M_3^1 = \begin{pmatrix} 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 0 \end{pmatrix}.$$

The largest matrix dimension among the matrices M is called *bond dimension.* In the GHZ example above, the dimensions are 1×2, 2×2 and 2×1, so the bond dimension is 2. In fact, the bond dimension for the n-qubit GHZ state for any positive integer n is 2, so that the total number of matrix entries that define the MPS is roughly $2 \cdot 2n$, which is much less than the 2^n entries that are needed when representing the GHZ state as state vector.

Equivalently to the matrix collection above, we can write an n-qubit MPS as a collection of 2 order-2 and $n - 2$ order-3 tensors, whose edge connection structure forms a line where the order-2 tensors are at the line's two endpoints. Contracting all connections then yields the quantum-state vector. The tensors are obtained

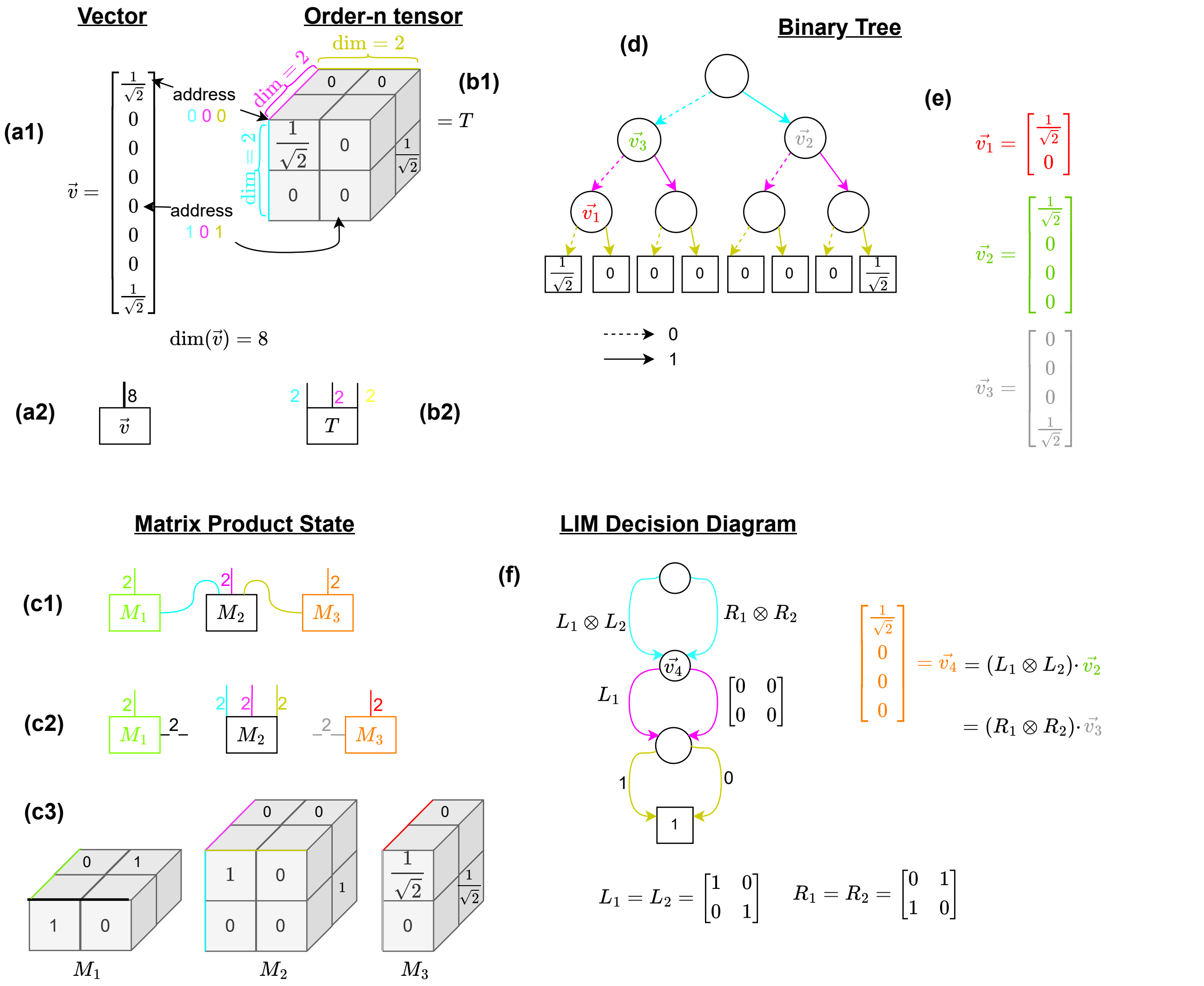
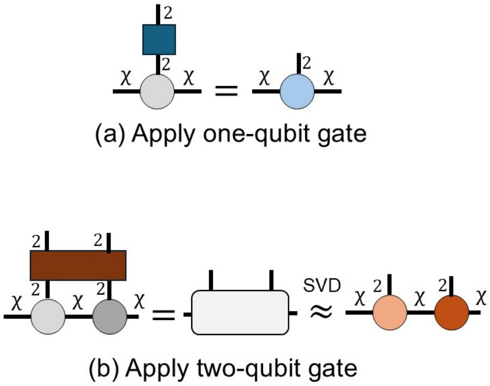


Figure 4: Various data structures representing the 3-qubit GHZ state. (a1) As vector of $2^3 = 8$ entries. Each entry is labeled a 3-bit address. The 8 entries can also be stacked differently as is done in the order-3 tensor T (b1). The order-3 tensor T has dimensions $2 \times 2 \times 2$, yielding precisely the $2^3 = 8$ different addresses. (b2) graphically depicts T as a node labeled with the letter 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 as a node with a single leg 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 and M_3 (c3) 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 Sec. A.3). (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 matrices. For LIMDD, the factor $\frac{1}{\sqrt{2}}$ that should be multiplied with each entry is kept track of separately (not displayed).

by stacking the two MPS matrices for qubit $j \in \{1, 2, \ldots, n\}$ such, that they form a single tensor. For the 3-qubit GHZ state example above, the 3 tensors (M_1, M_2, M_3) are given in Fig. 4(c3), and have dimensions 2×2, $2 \times 2 \times 2$ and 2×2.

Simulating [165] with MPS starts by constructing the MPS for the canonical input state, which has bond dimension 2 for any number of qubits n. To update an MPS of bond dimension χ after a single-qubit gate on qubit j, we note that the gate is an order-2 tensor (i.e. a matrix) of dimensions $d_{\text{in}} = 2$ and $d_{\text{out}} = 2$. Next, one adds the tensor graphically and connects the d_{in} edge to the size-2 edge of the j-th tensor in the MPS, see Fig. 5(a). Then, one contracts the connection, which updates the j-th tensor of the MPS only. The contraction requires accessing all $O(\chi^2)$ entries of the tensor, yielding a total time complexity of $O(\chi^2)$ [165].

Applying a two-qubit gate on adjacent qubits starts similarly and requires contracting tensors into an intermediate order-4 tensor, which has time complexity $O(\chi^2)$. See Fig. 5(b). To bring this back to two order-3 tensors, the Singular Value Decomposition (SVD) is applied. The bond dimension can increase at most quadratically during this process, i.e. from χ to χ^2. However, one can choose at this point to make the simulation *approximate* by dropping the terms with small contribution (i.e. small singular value) in order to reduce the bond dimension. Another approach at approximate simulation is to fix the maximal bond dimension χ and only keep the largest χ singular values. The SVD step dominates the cost of the two-qubit operation, and has a complexity of $O(\chi^3)$.



(a) Apply one-qubit gate

(b) Apply two-qubit gate

Figure 5: Applying gates to qubits in MPS for a fixed (capped) bond dimension χ. (b) A two-qubit gate can increase the bond dimension quadratically; by artificially reducing this back to χ, the output state is approximated.

A.4 Other representations

A *tensor network* [125] is a generalization of MPS where the graph structure is not a line but a more complex graph, and individual tensors can be of higher order 2 or 3. Due to these relaxations, tensor networks can succinctly represent states which MPS cannot [76], at the cost of increased hardness of operations [124], and have proven powerful when simulating quantum circuits [79].

Next, *decision diagrams*, originally developed for reasoning about binary functions, have also been applied to the quantum domain [155]. Decision diagrams can be thought of as a compressed version of a binary decision tree of height n, where the 2^n leaves contain the entries of the quantum-state, see Fig. 4(d,e). The compression is homomorphic, i.e. operations (such as quantum gates) can be applied on the decision diagram without ever having to unfold to the full tree. Among the most succinct yet tractable types are Local-Invertible-Map Decision Diagrams (LIMDDs), which achieve the compression by noting that nodes in the tree, which correspond to smaller vectors, need not be stored separately if they can be mapped to each other by simple (invertible) matrices [168]. See Fig. 4(f). LIMDDs can succinctly represent and simulate a range of relevant states and circuits [168, 169].

The *ZX-calculus* [162] 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 two 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 rewrite rules and a decomposition which expresses the entry as a linear combination of smaller diagrams which are easier to directly evaluate.

Finally, there are representations of quantum states other than the ones mentioned in the main text, such as the extended stabilizer [27] and matchgate [116] formalisms.

Appendix B PRELIMINARY FINDINGS

In this section, we first explain in Sec.B.1 how we represent quantum states in the relational model. Then, in Sec. B.2, we analyze the space and time complexities of the relational representation, comparing it with some of the existing quantum data representations used in our experiments. In Sec. B.3 and B.4, we further detail the experimental setup and report preliminary experimental results.

Both our analytical and experimental evaluations are illustrated using circuits for producing three quantum states: the W state and GHZ states, which found numerous applications in secure communication [101, 114, 115, 144] and quantum metrology [121], as well as a the output state of a circuit only containing the Quantum Fourier Transform (QFT) gate, a central operation in the famous quantum algorithm by Shor for factoring large integers. Fig. 6 shows the states as well as a circuit for producing them.

B.1 Relational representation for quantum states

We already have seen different representations of quantum states in Section A. Here, we describe a simple relational representation based on tensors (Section A.2). In particular, we propose a *sparse* relational encoding of tensors, only storing entries with non-zero values.[6] More precisely, to efficiently represent states using RDBMS, given an n-qubit state $|\psi\rangle$, we encode the non-zero amplitudes and their corresponding basis state indices in a relation R with arity $n+2$, as follows:

$$R(s_1, s_2, \cdots, s_n, r, c), \text{ for } s_k \in \{0,1\}, k \in [1,n], r \in \mathbb{R}, c \in \mathbb{R},$$

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 c are the real and imaginary parts, respectively, of the (possibly complex) amplitude associated with the corresponding *non-zero* basis state. Both r and c are real-valued.

Hence, a tuple $t \in R$ is a tuple representing one non-zero basis state and its amplitude. We remark that the total number of tuples in this representation of $|\psi\rangle$ is $\text{nnz}(|\psi\rangle)$, where $\text{nnz}(|\psi\rangle)$ denotes the number of non-zero amplitudes in the quantum state $|\psi\rangle$.

[6] Similar to [20], our relational representation is based on encoding general tensor expressions in a relation schema comprising tensor indices and values. In addition, we extend this representation to incorporate quantum-specific semantics. i.e., treating an n-qubit state as an order-n tensor, and storing only non-zero probability amplitudes. With simple transformations (merging multiple column values into a single text field), our representation can be converted into the qubit state representation used in [158].

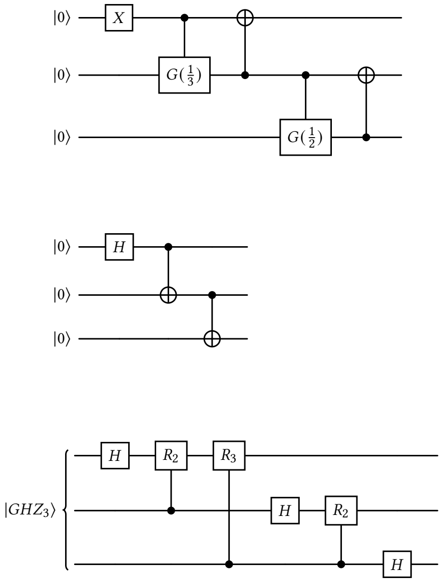
State

$$|W_n\rangle = \frac{1}{\sqrt{n}}\big(\; |100\ldots00\rangle + |010\ldots00\rangle + \cdots + |000\ldots01\rangle \big)$$

$$|GHZ_n\rangle = \frac{1}{\sqrt{2}}\left(\underbrace{|000\ldots0}_{n\text{ times}}\rangle + |111\ldots1\rangle\right) = \begin{pmatrix}\frac{1}{\sqrt{2}}\\ 0\\ 0\\ \vdots\\ 0\\ \frac{1}{\sqrt{2}}\end{pmatrix}$$

$$\begin{aligned} |QFT_n\rangle \quad &= \frac{1}{\sqrt{2^{n+1}}}\Big[\quad (1+\omega^0)\,|00\ldots00\rangle \\ &\qquad +(1+\omega^{-1})\,|00\ldots01\rangle \\ &\qquad +\ldots \\ &\qquad +(1+\omega^{-(2^n-1)})\,|11\ldots11\rangle\,\Big] \\ \text{where} \quad &\omega := e^{-2\pi i/2^n} = \cos(2\pi/2^n) + i\sin(2\pi/2^n) \end{aligned}$$

Circuit



Used gates

$$X = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$$

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

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

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

$$R_k = \begin{pmatrix}1 & 0\\ 0 & e^{2\pi i/2^k}\end{pmatrix}$$

Figure 6: Three n-qubit quantum states, with circuits that prepare them (the figure gives the $n = 3$ quantum circuit as example). Depicted matrices represent gates. A thick dot with line indicates a controlled-gate: the gate is only performed on the target qubit if the address bit of the control qubit reads 1. (Top) W state [49], the uniform superposition of n-bit strings with Hamming weight 1. The W state is sparse enough since the number of non-zero amplitudes in the standard basis is n but there are 2^n component. Circuit from [41], with total gate count $2n - 1$, consisting of $n - 1$ controlled-$G(p)$ gates, $n - 1$ $CNOT$-gates and one additional X gate. (Middle) GHZ state [68], whose vector is sparse as it only constains two nonzero entries. Its preparation requires entangling all qubits with each other. We use a circuit with n gates: a single Hadamard and $n - 1$ controlled-NOT (CNOT) gates [41]. (Bottom) The output state, denoted here as $|QFT_n\rangle$, of the n-qubit Quantum Fourier Transform [122] (QFT) gate, with the n-qubit GHZ state (here we use the 3-qubit GHZ state $|\text{GHZ}_3\rangle$) as input.

Example B.1. In Fig. 7, we illustrate the relational representation of the W, GHZ and QFT states. These states, their circuits, and the matrix representations of the gates used are illustrated in Fig. 6.

(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 amplitudes. It is represented by three tuples shown in Fig. 7a. For instance, the tuple $(0, 0, 1, \frac{1}{\sqrt{3}})$ corresponds to the basis state $|001\rangle$ with its amplitude $\frac{1}{\sqrt{3}}$.
(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 amplitudes. It is represented by two tuples in Fig. 7b.
(3) The output of the 3-qubit QFT circuit, shown in Fig. 6, when given the GHZ state $\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$ as input, is $|\psi_3\rangle = \frac{1}{4}\bigg(2\,|000\rangle + (1+i)\,|010\rangle + (1-i)\,|011\rangle + (1+\frac{1}{\sqrt{2}}+\frac{i}{\sqrt{2}})\,|100\rangle + (1-\frac{1}{\sqrt{2}}-\frac{i}{\sqrt{2}})\,|101\rangle + (1-\frac{1}{\sqrt{2}}+\frac{i}{\sqrt{2}})\,|110\rangle + (1+\frac{1}{\sqrt{2}}-\frac{i}{\sqrt{2}})\,|111\rangle\bigg)$. We depict its relational representation in Fig. 7c consisting of eight tuples.

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

(a) 3-qubit W state

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

(b) 3-qubit GHZ state

s_1	s_2	s_3	r	c
0	0	0	$\frac{1}{2}$	0
0	1	0	$\frac{1}{4}$	$\frac{1}{4}$
0	1	1	$\frac{1}{4}$	$-\frac{1}{4}$
1	0	0	$\frac{1}{4} + \frac{1}{4\sqrt{2}}$	$\frac{1}{4\sqrt{2}}$
1	0	1	$\frac{1}{4} - \frac{1}{4\sqrt{2}}$	$-\frac{1}{4\sqrt{2}}$
1	1	0	$\frac{1}{4} - \frac{1}{4\sqrt{2}}$	$\frac{1}{4\sqrt{2}}$
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 7: Relational representations of different quantum states.

In summary, our relational representation is designed to compactly encode sparse quantum states, significantly reducing storage requirements compared to the trivial representation, which requires 2^n tuples.

State $\vert\psi\rangle$	Order-n tensor (Baseline I)	Relational representation (RDBMS solutions)	MPS (Baseline II)
General state	$O(2^n)$	$O(n \cdot \mathrm{nnz}(\vert\psi\rangle))$	$O(n\chi^2)$
W$_n$ State	$O(2^n)$	$O(n^2)$	$O(n)$
GHZ$_n$ State	$O(2^n)$	$O(n)$	$O(n)$
QFT$_n$	$O(2^n)$	$O(n \cdot 2^n)$	$O(n\chi^2)$

Table 1: Space complexity comparison of different representations of state $|\psi\rangle$. Here, n is the number of qubits, $nnz(|\psi\rangle)$ denotes the number of non-zero probability amplitudes in the state $|\psi\rangle$, and the MPS bond dimension χ is a fixed constant that one chooses oneself, potentially making the representation approximate.

B.2 Complexity Analysis

We compare the space and time complexity of quantum computations across three representations used in our experiments. This section first addresses space complexity and then provides an overview of time complexity. The representations considered are:

- **NumPy Tensor Format:** A flattened row-major tensor representation. This approach avoids explicitly storing tensor indices, as elements are accessed using stride values along each dimension. The space complexity is measured in terms of the total number of entries.
- **Relational Tensor Format:** As just defined, this representation stores indices alongside non-zero values. The space complexity is measured in terms of number of non-zero values times the space per entries ($n + 2$).
- **Matrix Product State (MPS):** A decomposition of tensors with a bond dimension χ, where each component is stored in NumPy tensor format (See also Section A.3). The space complexity is the sum of the space complexity of these components. For a given state $|\psi\rangle$, we denote the smallest bond dimension, minimized over all MPS that represent $|\psi\rangle$, as $\chi_{\text{exact}}(|\psi\rangle)$. We fix χ to a constant throughout the simulation, which makes the simulation approximate if for some intermediate quantum state $|\psi\rangle$ in the circuit computation it holds that $\chi < \chi_{\text{exact}}(|\psi\rangle)$.

Space Complexity. As usual, we evaluate space complexity in terms of the size of the largest tensor encountered during computation. Let n represent the number of input qubits. A summary of the results is presented in Table 1. It is known that general quantum computations can be carried out using local gates, i.e., gates only affecting a constant number of qubits. We assume local gates throughout this section.

▷ *General Case.* For arbitrary quantum computations,space consumption depends on the representation of an arbitrary quantum state $|\psi\rangle$:

- For NumPy tensors, space complexity is $O(2^n)$, as it stores all 2^n elements of the tensor. Indeed, the application of local gates can be done in $O(2^n)$ space in as this corresponds to multiplication with a matrix of fixed (constant) dimensions.
- For the relational representation, only non-zero values are stored along with their indices, resulting in $O(n \cdot \mathrm{nnz}(|\psi\rangle))$, where $\mathrm{nnz}(|\psi\rangle)$ denotes the number of non-zero elements. Also here, locality of gates ensures that the constant matrix multiplication is carried by a join requiring $O(2^n)$ space.
- For MPS, with bond dimension χ, the representation includes two matrices of size $2 \times \chi$ and $n - 2$ tensors of size $\chi \times 2 \times \chi$. Stored in NumPy format, the space complexity is $O(n \cdot \chi^2)$.

▷ *Examples: W, GHZ, and QFT States.* For all three examples one can verify that the space complexity is dominated by the space complexity of their final quantum state. The number of non-zero elements for these states are $\mathrm{nnz}(|W_n\rangle) = n$ and $\mathrm{nnz}(|\mathrm{GHZ}_n\rangle) = 2$ and $\mathrm{nnz}(|\mathrm{QFT}_n\rangle) = 2^n$. Moreover, the W and GHZ states can be represented exactly with bond dimension $\chi_{\text{exact}} = 2$ [130, 169]. As a consequence, for these three states:

- For NumPy tensors, the space complexity remains $O(2^n)$ as sparsity is not leveraged.
- For the relational representation, the general space complexity is $O(n \cdot \mathrm{nnz}(|\psi\rangle))$, so becomes $O(n^2)$ for $|W_n\rangle$ and $O(n)$ for $|\mathrm{GHZ}_n\rangle$, by leveraging sparsity but at the cost of storing indices. For the n-qubit QFT state, it is $O(n \cdot 2^n)$.
- For MPS, as we fix the bond dimension χ, the tensors are all of size $O(\chi^2)$, yielding a combined space complexity of all n tensors of $O(n\chi^2)$. However, the W state and GHZ states both have constant bond dimension ($\chi_{\text{exact}} = 2$), independently of the number of qubits n, so the space complexity is $O(n)$ for these two states.

▷ *Remarks.* This analysis demonstrates the advantages of relational representations over standard numerical formats like NumPy, particularly for sparse quantum states. These advantages are evident in sparse quantum circuits, where the sparsity of the quantum state is preserved, as in state preparation tasks [107]. Currently, the relational representation does not utilize MPS factorization. However, each MPS component could naturally be represented relationally. Similarly, there are connections to treewidth of quantum circuits that could be integrated [108]. Developing more advanced compact representations, leveraging recent work on such representations (see Section A), represents a key research challenge.

Time complexity. The time complexity of simulating general quantum circuits is a complex problem and depends on many factors, including the number of gates, circuit depth, type of gates, circuit length, and the graph structure of the circuit. For our purposes, and in particular for our experiments, it suffices to provide a comparison of the time complexity of our three example circuits: The W state and GHZ state preparation circuits, and QFT circuits from Fig. 6. Our analysis is summarized in Table 2.

▷ *W state.* An n-qubit W_n state preparation circuit has one X gate, and $n - 1$ controlled $G(p)$ gates and $n - 1$ CNOT gates. As such, the overall number of gates is $O(n)$.

- For NumPy tensors, the X gate is a single-qubit gate represented as a 2×2 matrix. Both the controlled $G(p)$ and the CNOT are two-qubit gates are represented as order-4 tensors of size $2 \times 2 \times 2 \times 2$.

State $\|\psi\rangle$	**Order-n tensor** **(Baseline I)**	**Relational representation** **(RDBMS solutions)**	**MPS** **(Baseline II)**
W State	$O(n\cdot 2^n)$	$O(n^2)$	$O(n)$
GHZ State	$O(n\cdot 2^n)$	$O(n)$	$O(n)$
QFT	$O(n^2\cdot 2^n)$	$O(n^3\cdot 2^n)$	$O(n^3\chi^3)$

Table 2: Time complexity comparison for the circuits from Fig. 6 for producing state $|\psi\rangle$. Here, n is the number of qubits, and the MPS bond dimension χ is a fixed constant that one chooses oneself, potentially making the representation approximate.

Applying each single- or two-qubit gate requires $O(2^n)$ steps. Given that there are $O(n)$ gates, the time complexity is $O(n\cdot 2^n)$.

- For the relational representation, we can see that the application of the X gate keeps the number of non-zero elements invariant. In contrast, for the W-state preparation circuit, one can derive that that after the k-th controlled-NOT gate, the state is
$$\sqrt{\frac{1}{n}}\sum_{x\in F_k}|x00\dots0\rangle+\sqrt{\frac{n-k}{n}}\,|\underbrace{00\dots0}_{k\text{ zeroes}}100\dots0\rangle \qquad (1)$$
where F_k is the set of the first k bitstrings $10\cdots0, 010\cdots00, \dots$ of length $k{+}1$ with hamming weight equal to 1. This state has $k{+}1$ nonzero entries. The controlled-$G(p)$ gate before each CNOT gate does not change that number. Since applying a single- or two-qubit gate on state $|\phi\rangle$ takes at most $O(\text{nnz}(|\phi\rangle))$ time, and since the W-state circuit has n single- or two-qubit gates, we obtain a time complexity of $O(n^2)$.
- For MPS, we recall from Sec. A.3 that the time complexity for a single two-qubit gate is most $O(\chi^3)$. As we fix χ, and there are n gates, an upper bound to the time complexity is $O(n\cdot\chi^3)$. However, the intermediate states in the circuit as given in eq. (1) have *exact* bond dimension 2, independently of n or k. We thus conclude that the MPS time complexity for simulating the W state circuit is $O(1)$ per gate, and as there are $O(n)$ gates in the circuit, the total time complexity is $O(n)$.

▷ *GHZ state.* For the n-qubit GHZ$_n$ state circuit, we note that after the k-th CNOT, the state is
$$\frac{1}{\sqrt{2}}|00\dots0\rangle+\frac{1}{\sqrt{2}}|\underbrace{11\dots1}_{k+1\text{ times}}00\dots0\rangle$$
which has exact bond dimension 2, making its MPS time complexity similar to that of the W state. Since this intermediate state has only 2 nonzero entries, independently of k or n, updating the relational encoding upon applying a CNOT is done in constant time, making the time complexity $O(n)$ as there are $O(n)$ gates. The time complexity analysis for the NumPy tensor format is similar to the W_n state.

▷ *QFT.* The QFT circuit consists of $O(n^2)$ single- and two-qubit gates, so the time complexity for NumPy tensors becomes $O(n^2\cdot 2^n)$. For the relational representation, the time complexity scales similar to the NumPy tensor approach except for an additional factor n for the indices. That is, its time complexity is $O(n^3\cdot 2^n)$. For MPS, a single-qubit gate update runs in time $O(\chi^3)$, and so does the two-qubit gate as long as the qubits are adjacent. If this is not the case, as in the QFT circuit, additional SWAP gates are needed to bring the two qubits next to each other. For the QFT circuit, at most $O(n)$ SWAP gates are needed to make each gate act on adjacent qubits, increasing the gate count to $O(n^3)$, and hence the time complexity to $O(n^3\chi^3)$. We expect that χ_{exact} is constant for the $|QFT_n\rangle$ state, which might improve the bound to $O(n^3)$ when the bond dimension χ used in simulation is chosen $\chi\leq\chi_{\text{exact}}$, but we consider computing it exactly out of scope for this analysis.

B.3 Experiment setting

In this section, we explain our experimental setting. We report our 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 representations:

- *Baseline I*: tensor network representation and tensor contractions are performed over order-n tensor, where n is the number of qubits. This Baseline used the NumPy tensor format described in Section B.2.
- *Baseline II*: Matrix Product State (MPS) representation, where again each component is stored in NumPy format (see again Section B.2).

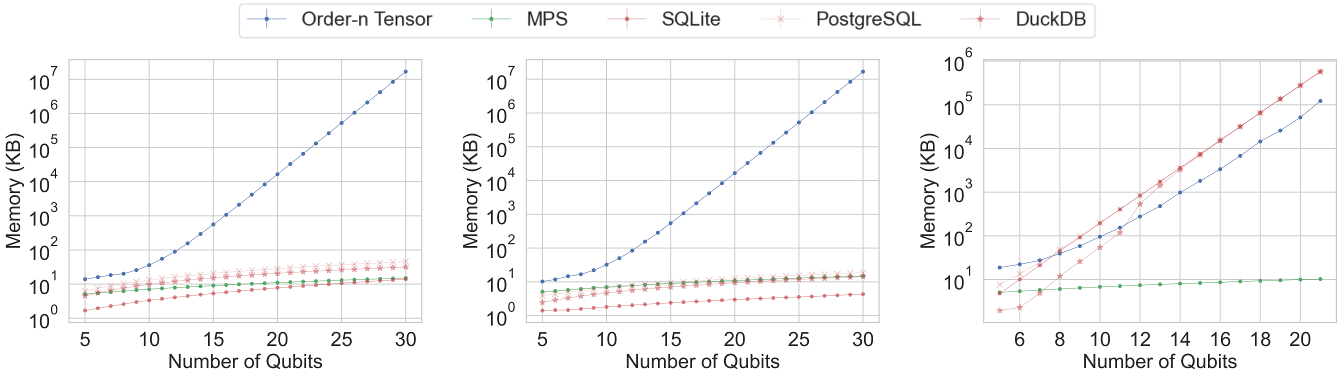
Both baselines were developed in Python 3.13.0 and NumPy 2.1.3, widely used linear algebra libraries for classical simulation.

- *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 made publicly available online.[7] 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[8] 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.

[7] https://github.com/infiniData-Lab/Quantum

[8] https://pypi.org/project/opt-einsum/



(a) W state (sparse circuit)

(b) GHZ state (sparse circuit)

(c) QFT (dense circuit)

Figure 8: 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. For sparse circuits (W and GHZ), RDBMS solutions demonstrate significantly lower memory usage than the order-n tensor baseline, with SQLite showing a slight advantage over the MPS baseline. For the dense circuit QFT, the memory usage of RDBMS solutions grows comparably to or exceeds the order-n tensor baseline, indicating the need for improvements in handling dense tensor computations.

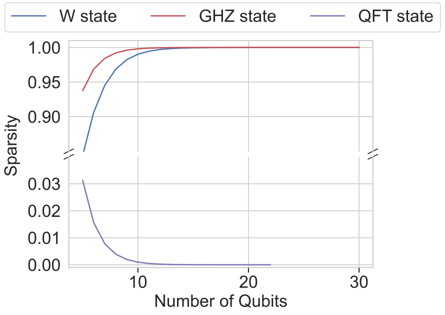


Figure 9: Sparsity ($s = 1 - \frac{\mathbf{nnz}(|\Psi\rangle)}{2^n}$) comparison for GHZ and W state preparation circuits and QFT circuit, where nnz($|\Psi\rangle$) denotes the number of non-zero probability amplitudes in the state $|\Psi\rangle$, n is the number of qubits and 2^n is total number of possible states.

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 $|0\rangle$ — 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 SUM operations over the shared index in SQL, with tensors as relational tables. The resulting tensor is grouped by the output indices. 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 followed by a *CNOT*-gate to a two-qubit system would look as follows with T0, T1 and T2 as the intermediate states and CX and H as the corresponding gates:

```
WITH T0(s0, r) AS (
    VALUES
    (0, 1.0)),
   H(i, j, r) AS (
    VALUES
    (0, 0, 0.7071067811865475),
    (0, 1, 0.7071067811865475),
    (1, 0, 0.7071067811865475),
    (1, 1, -0.7071067811865475)),
   CX(i, j, k, l, r) AS (
    VALUES
    (0, 0, 0, 0, 1.0),
    (0, 1, 0, 1, 1.0),
    (1, 0, 1, 1, 1.0),
    (1, 1, 1, 0, 1.0)),
T1 AS (
SELECT CX.i AS i, CX.j AS s1, CX.k AS s0,
SUM(CX.r * T0.r) AS r FROM CX, T0 WHERE CX.l=T0.s0
GROUP BY CX.i, CX.j, CX.k),
T2 AS (
SELECT H.i AS s1,
SUM(H.r * T0.r) AS r FROM H, T0 WHERE H.j=T0.s0
GROUP BY H.i) SELECT T1.s1 AS s1, T1.s0 AS s0,
SUM(T2.r * T1.r) AS r FROM T2, T1 WHERE T2.s1=T1.s1
GROUP BY T1.s1, T1.s0 ORDER BY s1, s0
```

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 runtime (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?

B.4.1 Memory Usage. Fig. 8 compares the memory usage of RDBMS solutions and two baselines.[9] 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., SQLite, PostgreSQL, and DuckDB, demonstrate significantly lower memory consumption in sparse scenarios (W state and GHZ state preparation circuits). 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.

An interesting observation from Fig. 8a and 8b 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 small advantage over the MPS baseline. Fig. 8b shows that this advantage of RDBMS solutions is even more evident in the GHZ state, which is sparser than the W state (cf. Fig. 9). 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. 8c, 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. As shown in Fig. 9, the final state of QFT is significantly less sparse compared to the W and GHZ states. Besides the final state, the intermediate states generated during QFT circuit simulations are also dense.

We further explore the scalability advantages of RDBMS solutions compared to the general baseline using order-n tensor representation. Fig. 10 compares the maximum number of qubits that can be simulated by each method under a 2.0 GB memory limit. For the W state preparation circuit in Fig. 10a, the RDBMS solutions (SQLite, PostgreSQL, DuckDB) significantly outperform the order-n tensor baseline, in the best case (DuckDB), supporting over 53017 qubits compared to just 17 qubits for the baseline. This represents a 3118× improvement in scalability. As shown in Fig. 9, GHZ state preparation circuit, in general, involves even sparser tensors than W state preparation circuit. For GHZ state preparation circuit, SQLite performs the best and simulates up to $1.4 \cdot 10^7$ qubits, compared to 17 for order-n tensor baseline, yielding an $8.3 \cdot 10^5\times$ improvement.[10] These results demonstrate the substantial memory efficiency and scalability advantages of RDBMS solutions in handling sparse circuits. For the dense circuit QFT, RDBMS solutions simulate up to 14 qubits, which is comparable to the 16 qubits supported by order-n tensor baseline with a reduction of 14%.

B.4.2 Runtime. Figure 11a compares the runtime of the RDBMS solutions with the baselines. The observations are, in general, similar to memory usage. RDBMS solutions outperform the order-n tensor baseline for sparse circuits (W and GHZ state preparation) but not for the dense circuit QFT. This is consistent with the time complexity analysis in Table 2. Notably, runtime is influenced by many factors including implementation details. For instance, for W state and GHZ state preparation circuits, the SQLite solution shows runtime advantages over both baselines, while the benefits of PostgreSQL and DuckDB over the order-n tensor baseline, become evident only for larger qubit counts (above 23 qubits). These results suggest that to further enhance performance, the optimizations might also need to tailored to individual database systems. As shown in figure 11c, the QFT circuit, which involves dense tensor computation, exhibits a different trend. Beyond 12–14 qubits, all RDBMS solutions become slower than both baseline, which is as expected from Table 2. Moreover, the baselines are implemented in NumPy, which handles dense tensor operations more efficiently than the RDBMS solutions. This indicates the need for further improvements in both representation and implementation to better simulate dense circuits using RDBMSs.

Summary. For Q1, pushing simulation workloads to RDBMS appears to be a promising approach. For circuits with sparse tensors (W and GHZ), RDBMS solutions demonstrate significant memory and time efficiency, enabling simulations of much larger qubit counts compared to the order-n tensor baseline, while being competitive with the state-of-the-art MPS baseline. Further optimization opportunities remain for even better performance. For circuit with dense tensors (QFT), advanced relational representations (beyond Sec. B.1), improved system implementations and more sophisticated optimizations are necessary to address current limitations.

Takeaways:

(1) ***T1 RDBMS for sparse circuits:***
Relational representations are efficient and scalable for quantum circuits involving sparse tensor computations. They are competitive with, and may even outperform, state-of-the-art representations like MPS in specific scenarios.

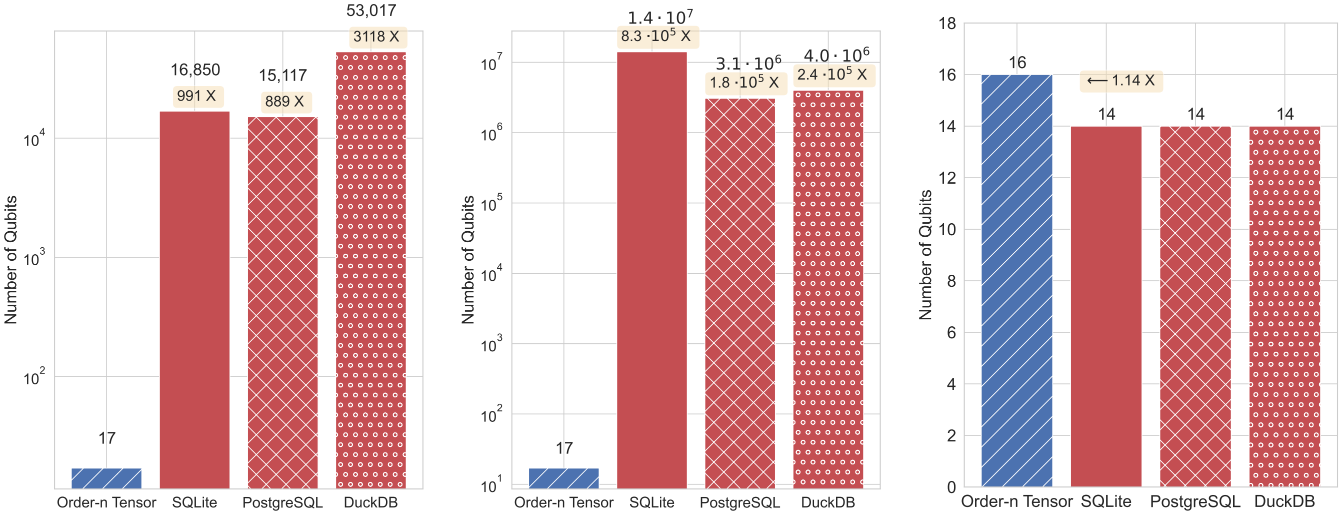
(2) ***T2 Opportunities for Optimization:***
Exciting research opportunities remain, to optimize existing RDBMSs for simulation workload.

B.4.3 Input query generation. The input to a simulator is the quantum circuit to be simulated. For RDBMS-based simulators, input circuits can be represented as SQL queries. Our preliminary experiments identified the following research gap in the state-of-the-art approach [20] for generating the input circuit SQL queries:

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

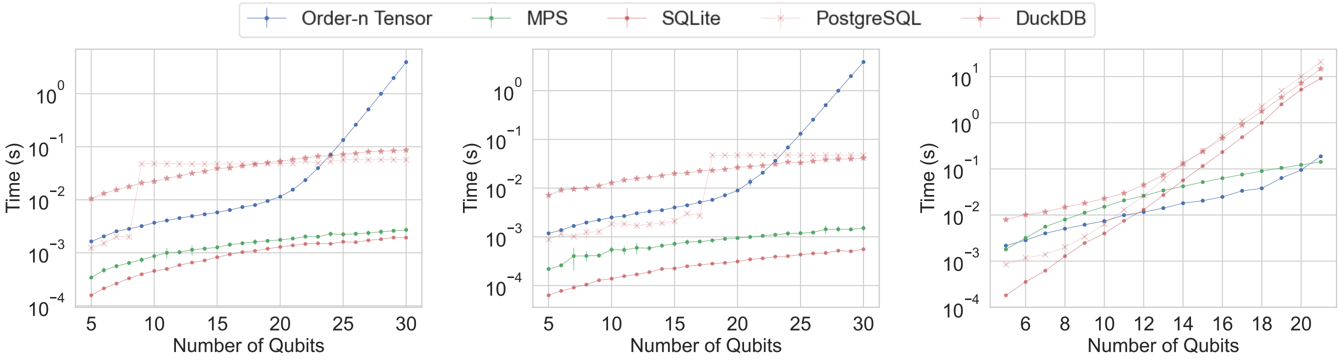
[9] In all QFT-related experiments in this section, we were limited to 21 qubits due to the substantial time required to complete 100 runs, and the memory limit of our hardware.

[10] The exact qubit counts are 14107345, 3079324, and 4000862 for SQLite, PostgreSQL, DuckDB, respectively.



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

Figure 10: Comparison of the simulated numbers of qubits for RDBMS solutions and the order-n baseline under a 2.0 GB memory limit. The y-axis is in $\log_{10}$-scale for W and GHZ state preparation circuits. RDBMS-based approaches exhibit significant scalability in sparse circuits, simulating up to 16,850, 15,117, and 53,017 qubits on average for W states—yielding improvements of $991\times$, $889\times$, and $3,118\times$, respectively. For GHZ states, RDBMS solutions simulate $1.4 \cdot 10^7$, $3.1 \cdot 10^6$, and $4.0 \cdot 10^6$ qubits, achieving enhancements of $8.3 \cdot 10^5\times$, $1.8 \cdot 10^5\times$, and $2.4 \cdot 10^5\times$, respectively. For the dense circuit QFT, RDBMS solutions simulate up to 14 qubits, slightly below the 16 qubits supported by the order-n tensor baseline, indicating the need for further improvements.



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

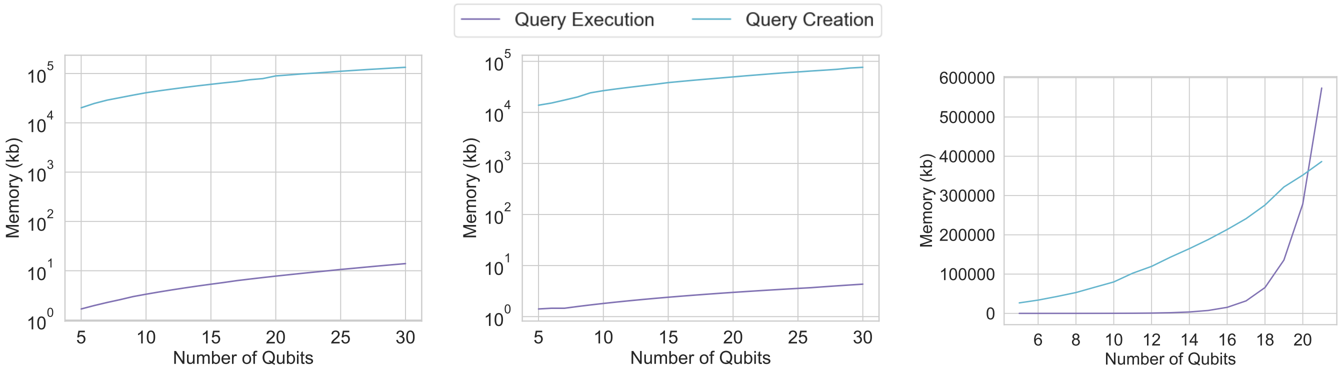
Figure 11: Runtime comparison of three RDBMS-based solutions (red) against general order-n tensor baseline (blue) and SotA MPS baseline (green). The y-axis is in $\log_{10}$ scale. The SQLite solution outperforms both baselines for sparse circuits. For the dense circuit QFT, NumPy-based baselines handle dense tensor computations more efficiently, indicating the need for further optimizations in RDBMS solutions.

(b) How good is the state-of-the-art SQL query generation method for quantum circuit simulation?

To generate the SQL queries for circuit simulation, the SotA approach [20] transforms tensors representing qubit states and gates—initially represented as NumPy arrays—into a relational representation using a Coordinate (COO) tensor format[11]. Furthermore, the tensor contraction paths also need to be translated into a sequence of join operations in SQL.

Fig. 12 compares the memory usage of SQL query generation using the approach in [20] against running the generated query. Across all cases, query generation incurs significant memory overhead. The contrast to query execution is more obvious for sparse circuits like W and GHZ state preparation, where the RDBMS solutions are more efficient. The overhead of query generation primarily arises from transforming tensors. Besides the number of qubits, the memory cost of tensor translation mainly depends on the number and variety of tensors representing gates. For instance, while the W and GHZ state circuits (Fig 6) share the same number of CONT gates, the W state circuit also includes a number of $G(p)$ gates,

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

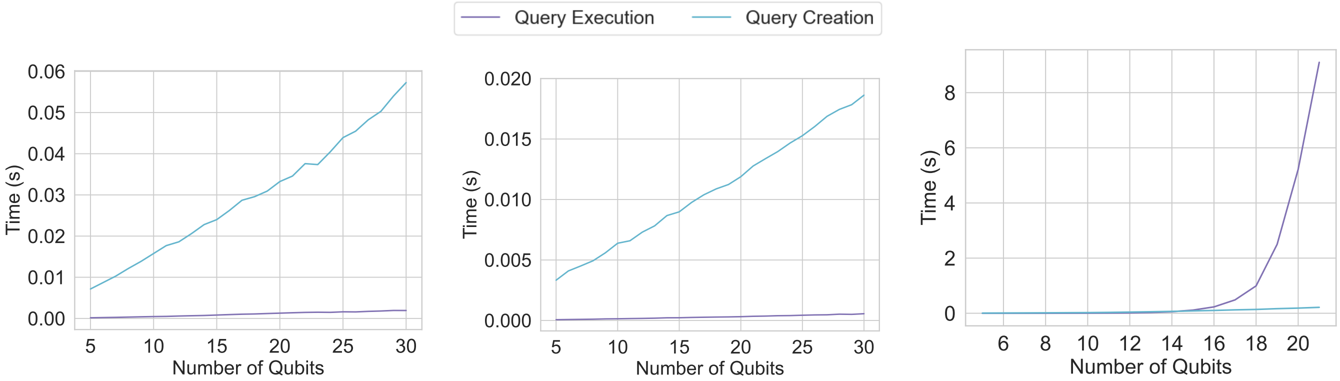


(a) W state (sparse circuit)

(b) GHZ state (sparse circuit)

(c) QFT (dense circuit)

Figure 12: Memory usage comparison of SQL query generation [20] vs. SQL query execution for SQLite. The y-axis in (a) and (b) uses $\log_{10}$-scale due to significant memory usage differences. Query generation incurs substantial memory overhead, in sharp contrast to query execution, especially for sparse circuits like W and GHZ states, where RDBMS solutions are more memory-efficient. This overhead results from tensor transformations and the translation of contraction paths into SQL join operations in [20].



(a) W state (sparse circuit)

(b) GHZ state (sparse circuit)

(c) QFT (dense circuit)

Figure 13: Comparison of SQL query generation time [20] vs. SQL query execution time for SQLite. Query generation requires more time than execution for sparse circuits like W and GHZ states, where RDBMS solutions are more efficient. Query generation incurs higher time costs due to the string construction proportional to the number of gates, along with overhead from tensor transformations and contraction path translation into SQL join operations in [20].

which require additional memory for their tensor translation, as seen in Fig. 12a and b. Moreover, the tensor contraction paths need to be translated into a sequence of JOIN operations in SQL. As the number of qubits increases, the gate count and circuit depth grow, further amplifying the overhead of contraction path translation.

Fig. 13 compares the query generation time using [20] with query execution time. SQL query generation involves constructing a string representation of the query, with the string length increasing proportionally to the number of gates in the quantum circuit, leading to higher time costs. 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 two steps introduces computational effort that grows significantly with the number of gates and circuit depth.

Summary. The state-of-the-art SQL query generation method [20] for quantum circuit simulation is inefficient, especially or sparse circuits like W and GHZ states. Possible improvement strategies include pre-storing the tensors in the database, eliminating the need for runtime translation and enabling direct access for quantum circuit operations. Future improvements, such as optimizing the contraction path generation process, could also reduce the query generation 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 offer a promising solution for analyzing simulations and quantum

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 like SQL, relational algebra, or tensor languages simplify data retrieval by providing intuitive, abstract constructs. These languages allow users to perform efficient and complex queries without requiring low-level programming, optimizing the interaction with large datasets.
Consistency, Recovery, and Transactions	Manual effort 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

experiments, since NoSQL databases are well-suited for managing unstructured data with their scalability and flexible schemas. A forward-thinking extension of Q1 in Sec. 4.2 is to choose most suitable databases for simulation. For example, TileDB [5], 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. Adapting array databases to simulation workloads remains an open challenge, requiring significant effort to enable in-database computation and query optimization for simulation workload and expand the their ecosystem to match the versatility of RDBMSs.

Moreover, 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 and could play a critical role in this context. Furthermore, for error correction techniques discussed in Sec. 4.3, exploring the recent effort of graph analytics [15, 106, 110, 171, 178] and graph databases [156] such as Neo4j[12], TigerGraph[13], might yield valuable insights.

In summary, 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 application.

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, 63, 93] 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. Please refer to our response in D2/R2.

4. Scalability. We have discussed the bottleneck of scalability without databases in Sec. 4.1. With Fig. 8 and 10, we identified the potential benefits of database-based approaches for memory-efficient simulations. Another promising direction is out-of-core simulation using databases [158]. 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

[12] https://neo4j.com/

[13] https://www.tigergraph.com/

that is significantly slower than in-memory processing. Recent advancements in out-of-core database techniques, such as sorting [95], aggregation [94], and joins, 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.