Towards Update-Dependent Analysis of Query Maintenance

XIAO HU, University of Waterloo, Canada
QICHEN WANG, Hong Kong Baptist University, China

This paper studies the hardness of maintaining self-join-free conjunctive queries over a dynamic database, where tuples can be inserted or deleted. The worst-case complexity of this problem under arbitrary updates has been well understood. It is known that most practical queries require $\Omega(\sqrt{|D|})$ maintenance time for each update to ensure $O(1)$-delay enumeration, barring a very restricted class of queries (known as "q-hierarchical" queries). Nonetheless, most real-world update sequences are not arbitrary, far away from the worst-case scenario; instead, they are so "nice" that queries can greatly benefit from their inherent structure in query maintenance. In this paper, we aim to understand the hardness of query maintenance under different update sequences, in particular, the insertion-only (or deletion-only), first-in-first-out (FIFO), arbitrarily worse sequences, as well as their "mixed" sequences. We first provide a comprehensive characterization of queries that can be maintained in $O(1)$ time for $O(1)$-delay enumeration over FIFO sequences. Then, we address mixed sequences, which may exhibit insertion-only or FIFO patterns on subqueries but lack a specific pattern in totality, and introduce a structural dichotomy for determining whether the input query can be maintained in $O(1)$ time for $O(1)$-delay enumeration over mixed sequences.

CCS Concepts: • **Theory of computation → Database query processing and optimization (theory)**.

Additional Key Words and Phrases: conjunctive query, insertion-only updates, FIFO updates, mixed updates

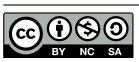
ACM Reference Format:
Xiao Hu and Qichen Wang. 2025. Towards Update-Dependent Analysis of Query Maintenance. *Proc. ACM Manag. Data* 3, 2 (PODS), Article 117 (May 2025), 25 pages. https://doi.org/10.1145/3725254

1 INTRODUCTION

Dynamic query processing is a challenging problem that has been extensively investigated [4, 7, 8, 11, 13, 16, 18–20, 22, 23, 25, 26], which studies how to maintain the query results over a dynamic database, where tuples can be inserted or deleted. Instead of computing query results from scratch each time, a common approach is to design a data structure that can be updated efficiently while the query answers can be retrieved with a delay guarantee. The worst-case complexity of this problem has been well understood [7, 16], but only a limited class of queries can admit an efficient index, and the large remaining queries only have a very expensive index. More importantly, the worst-case update sequence rarely occurs in practice. These observations motivate us to investigate more instance-dependent complexity of this problem, such as insertion-only and FIFO sequences.

Furthermore, practical update sequences usually exhibit patterns (insertion-only, FIFO, or arbitrary) over subqueries but lack a specific pattern in total. Let's look at the TPC-H benchmark [1], which has broad industry-wide relevance. There are 8 individual relations in the schema – NATION and REGION are public as well as static, which can be modeled as a special insertion-only sequence before any update from other relations; PART, SUPPLIER and CUSTOMER are insertion-only, as these entities are not deleted from database once inserted; in contrast, LINEITEM, ORDER, and

Authors' addresses: Xiao Hu, xiaohu@uwaterloo.ca, University of Waterloo, 200 University Ave W, Waterloo, Ontario, Canada, N2L 3G1; Qichen Wang, qcwang@hkbu.edu.hk, Hong Kong Baptist University, China.



2836-6573/2025/5-ART117
https://doi.org/10.1145/3725254

PARTSUPP are subject to frequent updates, which could be FIFO or even arbitrary. Consider a simplified query in this benchmark that finds all parts and supplier information:

$$\text{PART(PK)} \bowtie \text{PARTSUPP(PK, SK)} \bowtie \text{SUPPLIER(SK)}$$

From our previous knowledge [7], this query is very costly to maintain as it is non-q-hierarchical. However, if updates in PART and SUPPLIER are insertion-only, this query becomes easy to maintain, even if updates in PARTSUPP are arbitrary [25]. To bridge the gap between existing theory and practical applications, we aim for a more fine-grained analysis of maintaining CQs over mixed sequences that display specific patterns (insertion-only or FIFO) on sub-queries.

1.1 Problem Definition

Conjunctive Query. Let $\mathbb{R}$ be a database schema with m relations $R_1, R_2, \cdots, R_m$ and n attributes $\mathcal{V} = \{x_1, x_2, \cdots, x_n\}$. Each relation R_i is defined on a subset of attributes $e_i \subseteq \mathcal{V}$. Let $\text{dom}(x)$ be the domain of attribute x. Let $\text{dom}(X) = \prod_{x \in X} \text{dom}(x)$ be the domain of a subset of attributes X. Let D be a given instance of $\mathbb{R}$, and the corresponding instances of $R_1, \cdots, R_m$ be $R_1^D, \cdots, R_m^D$, where R_i^D is a finite set of tuples from $\text{dom}(e_i)$. When the context is clear, we drop the superscript and use R_i for relation and instance.

In this paper, we consider the class of *self-join-free conjunctive queries* (CQs) formally defined as[1]

$$Q := \pi_{\mathbf{y}} \left(R_1(e_1) \bowtie R_2(e_2) \bowtie \cdots \bowtie R_m(e_m) \right),$$

where $\mathbf{y} \subseteq \mathcal{V}$ denotes the *output attributes*. Let $\bar{\mathbf{y}} = \mathcal{V} - \mathbf{y}$ denote the *non-output attributes*. Each R_i in Q is distinct, i.e., the CQ does not have a self-join. If $\mathbf{y} = \{x_1, x_2, \cdots, x_n\}$, such a CQ is a *full join*, i.e., the natural join of underlying relations. For simplicity, we omit $\pi_{\mathbf{y}}$ for a full join. If $\mathbf{y} = \emptyset$, such a CQ is known as a *Boolean CQ*, which indicates whether there exists any result of the underlying join query. We also use a triple $(\mathcal{V}, \mathcal{E}, \mathbf{y})$ to represent Q, where $\mathcal{E} = \{e_1, e_2, \cdots, e_m\}$.

We use Figure 1 to illustrate the relationship between different classes of CQs to be discussed throughout the paper and defer their formal definitions to Section 2.

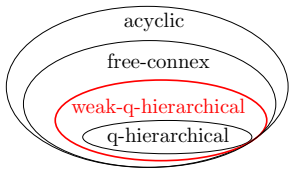


Figure 1. Classification of CQs.

Updates and Update Sequence. In the dynamic setting, we model each update as a quadruple $(t, s, +/\text{-}, R_i)$ for $s \in \mathbb{Z}$, indicating that tuple t is inserted into $(+)$ or deleted from (-) relation R_i at timestamp s. Let S be a sequence of updates ordered by their timestamp, where only a single update occurs at any given timestamp. An enumeration procedure may be invoked after any timestamp. Rather than specifying a separate preprocessing step, the initial database state may be simulated by using an insertion-only update sequence concatenated as the prefix of an arbitrary update sequence. Suppose the initial database contains n tuples and that the current timestamp is 0. Each tuple $t \in R_i$ will be assigned a distinct timestamp s within the range $[-n, -1]$. Consequently, for each tuple t, the update sequence includes the quadruple $(t, s, +, R_i)$ for each tuple t. For the remainder of this discussion, we assume the initial database is (logically) empty, with all pre-existing tuples inserted at negative timestamps.

Under set semantics, inserting t into R_i has no effect if R_i already contains t, and likewise, deleting t has no effect if t is not present in R_i. For a given update sequence S, the *lifespan* of a tuple t is an interval $[t^+, t^-]$, where t^+ denotes the timestamp when t is inserted, and t^- denotes the timestamp when t is deleted. For any tuple t not deleted by the end of S, we set $t^- = +\infty$ to indicate its persistence beyond the scope of the considered timestamp. Moreover, we treat two $+\infty$

[1]Hence, "self-join-free CQ" is simplified as "CQ" in the remaining of this paper.

values as incomparable, thereby ensuring that the single-update assumption is maintained. It is possible for the same tuple to be inserted and subsequently deleted multiple times. For simplicity, these instances are regarded as logically distinct tuples, with each one possessing a unique lifespan. These lifespans, corresponding to the same physical tuple, are assumed to be disjoint under set semantics. The dynamic database D is defined by the update sequence S in such a way that for every timestamp $s \in \mathbb{Z}$, the set of tuples t for which $s \in [t^+, t^-]$ constitutes a snapshot of D. The size of the dynamic database D is then defined as the maximum number of tuples that coexist at any given timestamp.

We are particularly interested in two classes of update sequences that are practically important:

- **First-in-first-out (FIFO)**. An update sequence S is *FIFO* if, for any pair of tuples t_1, t_2, if $t_1^+ < t_2^+$, then $t_1^- < t_2^-$ or $t_2^- = +\infty$, and *not FIFO* otherwise. FIFO sequences are commonly used in sliding-window or tumbling-window models applied to streaming data.
- **Insertion-only or Deletion-only.** An update sequence S is *insertion-only* if for every tuple t, $t^- = +\infty$, and *not insertion-only* otherwise. By definition, insertion-only sequences are inherently FIFO. Symmetrically, an update sequence S is *deletion-only* if for every tuple t, $t^+ < 0$, and *not deletion-only* otherwise. In subsequent discussions, we will focus on the insertion-only case, whereas an analysis of the deletion-only case is deferred to Appendix C.

For a CQ $\boldsymbol{Q} = (\mathcal{V}, \mathcal{E}, \mathbf{y})$ and an update sequence S, the *projection* of S to a subset $\mathcal{E}_1 \subseteq \mathcal{E}$ of relations is defined as $\pi_{\mathcal{E}_1} S = \{(*, *, +/-, R_e) \in S : e \in \mathcal{E}_1\}$, that is, it consists solely of those updates in S that pertain to relations from $\mathcal{E}_1$. We define a *pattern* $\sigma : 2^{\mathcal{E}} \to \{\text{insertion-only, FIFO, arbitrary}\}$ to characterize update sequences for $\boldsymbol{Q}$. An update sequence S is said to be *consistent* with the pattern σ, if for every subset $\mathcal{E}_1 \subseteq \mathcal{E}$ of relations, the projection $\pi_{\mathcal{E}_1} S$ adheres to the update behavior specified by $\sigma(\mathcal{E}_1)$. Furthermore, S is called a σ-sequence if it is consistent with σ. The projection of the pattern σ onto a subset $\mathcal{E}_1 \subseteq \mathcal{E}$ of relations is defined as $(\pi_{\mathcal{E}_1}\sigma) : 2^{\mathcal{E}_1} \to \{\text{insertion-only, FIFO, arbitrary}\}$, such that for every subset $\mathcal{E}_2 \subseteq \mathcal{E}_1$, $(\pi_{\mathcal{E}_1}\sigma)(\mathcal{E}_2) = \sigma(\mathcal{E}_2)$.

Enumeration. We focus on $O(1)$-*delay enumeration*, such that the time from the start of the enumeration to the first result, the time between any consecutive pair of results, and the time from the last result to the termination of the enumeration process, are $O(1)$. In this paper, we aim to understand the *maintenance complexity* of CQs over update sequences consistent to a given pattern σ, if targeting $O(1)$-delay enumeration. For simplicity, "$\boldsymbol{Q}$ can (resp. cannot) be maintained in α time over a set $\mathcal{S}$ of update sequences" should be translated as "there exists an index (resp. there doesn't exist an index) that can be maintained over an arbitrary update sequence $S \in \mathcal{S}$ in amortized α time per update, and support $O(1)$-delay enumeration for $\boldsymbol{Q}$ whenever needed".

Model of Computation. We use the standard RAM model under the uniform cost measure. For an input of size N, every register is assumed to have a length $O(\log N)$. Any arithmetic operation – such as addition, subtraction, multiplication, division – as well as the concatenation of the values of two registers, can be performed in $O(1)$ time. Moreover, retrieving the content of any register by its unique address is also achievable in $O(1)$ time.

1.2 Previous Results

Maintaining CQs under arbitrary updates. In 2017, two independent papers [7, 16] investigated the worst-case complexity of dynamically maintaining CQs under arbitrary updates. Their results establish the following dichotomy: Any *q-hierarchical* CQ can be maintained in $O(1)$ time under arbitrary updates, while for non-q-hierarchical CQs, a polynominal lower bound on the update time has been demonstrated, assuming the OMv and OuMv conjectures.

Theorem 1.1 ([7, 16]). *A CQ Q can be maintained in $O(1)$ time over arbitrary updates if Q is q-hierarchical, and cannot be maintained in $O\left(\sqrt{|D|}^{1-\epsilon}\right)$ time for arbitrary $\epsilon > 0$ otherwise, assuming the OMv and OuMv conjectures.*

Conjecture 1.2 (OMv Conjecture [15]). *The following problem cannot be solved in $O(n^{3-\epsilon})$ time for any constant $\epsilon > 0$: Given an $n \times n$ Boolean matrix M and a sequence of n-dimensional Boolean vectors $v_1, v_2, \cdots, v_n$, it is required to output Mv_i before seeing v_{i+1}, for every $i \in [n]$.*

Conjecture 1.3 (OuMv Conjecture [15]). *The following problem cannot be solved in $O(n^{3-\epsilon})$ time for any constant $\epsilon > 0$: Given an $n \times n$ Boolean matrix M and a sequence of n-dimensional Boolean vector pairs $(u_1, v_1), (u_2, v_2), \cdots, (u_n, v_n)$, it is required to output $u_i^T M v_i$ before seeing (u_{i+1}, v_{i+1}), for every $i \in [n]$.*

Later, this $\Omega(\sqrt{|D|})$ lower bound was matched for certain specific non-q-hierarchical CQs, such as the triangle [18] and the length-4 cycle [14]. Moreover, any free-connex CQ can be maintained in $O(|D|)$ time under arbitrary updates [16, 25]. Consequently, a $\sqrt{|D|}$-gap exists between the lower and upper bounds for general non-q-hierarchical CQs. In addition, the tradeoff between update time and enumeration delay has also been investigated for specific queries, including the triangle query [18] and hierarchical CQs [19]. However, the derived theoretical lower bounds typically require only the identification of a single hard instance among all possible update sequences, which results in particularly loose lower bounds for many simple update sequences. Recently, people have investigated the instance-dependent complexity of maintaining CQs based on the enclosureness of update sequences [26] for foreign-key acyclic joins. Building on this idea, Wang et al. [25] recently extended the notion of enclosureness to free-connex CQs, and demonstrated that such CQs can be maintained in time proportional to the enclosureness of the update sequence. The formal definition of enclosureness will be presented in Section 2. This concept constitutes a significant aspect of our dichotomy results, as it helps establish the optimality of our lower bounds.

Lemma 1.4 ([25], Lemma 6.6). *For a free-connex CQ Q with a free-connex join tree $\mathcal{T}$, there is an index that can be updated in $O(\lambda)$ amortized time over an update sequence of enclosureness λ under $\mathcal{T}$, while supporting $O(1)$-delay enumeration for Q.*

Enumerating CQs over static databases. Enumerating CQs over static databases can be simulated by an insertion-only sequence as follows: (i) the initial database is assumed to be empty; (ii) input tuples are inserted sequentially, one by one; and (iii) after all tuples have been inserted into the database, an enumeration procedure is invoked to generate the query results.

Lemma 1.5 ([16]). *For any CQ Q, if Q can be maintained in α time over insertion-only sequences, then for an arbitrary database D, an index can be built in $O(\alpha \cdot |D|)$ preprocessing time, from which all query results of $Q(D)$ can be enumerated with $O(1)$ delay.*

From Lemma 1.4, Wang et al. [25] also obtained the following results:

Corollary 1.6 ([25], Theorem 6.11). *A free-connex CQ Q can be maintained in $O(1)$ time over insertion-only update sequences.*

On the other hand, any lower bound for enumerating CQs over static databases implies a lower bound for maintaining CQs over insertion-only sequences. Bagan et al. [5] and Brault-Baron [9] showed that after $O(|D|)$ preprocessing time, a CQ Q over any static database D can be enumerated with $O(1)$ delay, if and only if Q is free-connex. Putting everything together, we have:

Theorem 1.7. *A CQ Q can be maintained in $O(1)$ time over insertion-only updates if Q is free-connex and cannot be maintained in $O(1)$ time otherwise, assuming the Boolean Matrix Multiplication, Triangle Detection, and HyperClique conjectures.*

Conjecture 1.8 (Triangle Detection conjecture [2]). *Given a graph with m edges, no algorithm can decide whether a triangle exists in $O(m)$ time.*

Conjecture 1.9 (Boolean Matrix Multiplication [24] Conjecture). *Given two Boolean matrices of size $n \times n$, no algorithm can compute their product in $O(n^2)$ time.*

Conjecture 1.10 (HyperClique Conjecture [21]). *Given a k-uniform hypergraph (for $k \geqslant 3$), no algorithm can decide whether a hyper-clique of size $k+1$ exists or not in $O(m)$ time, i.e., a set of $k+1$ vertices where every subset of size k forms a hyperedge, where m is the number of edges in the hypergraph.*

Enumerating CQs under other updates. In an independent work, Kara et al. [17] studied the enumeration of CQs over databases where some relations remain static while others undergo arbitrary updates. Their goal is to characterize the class of CQs for which an index can be maintained with $O(1)$ amortized update time while supporting $O(1)$-delay enumeration. They further refine this characterization by considering three cases based on whether the preprocessing time is restricted to linear, polynomial, or exponential. In contrast, our work investigates a more general setting where updates in each relation — or even a subset of relations — may follow different patterns, including insertion-only, FIFO, or arbitrary. However, we implicitly restrict the preprocessing time to be linear complexity. As the settings considered in these other works differ fundamentally from our framework, direct comparisons are not feasible. Meanwhile, Abo Khamis et al. [3] investigate the difference in the amortized update time for maintaining general queries under insertion-only versus arbitrary update sequences, an investigation that diverges from our focus on characterizing queries maintainable within which are different from our target for characterizing $O(1)$ amortized update time.

1.3 Our Results

Due to the inherent difficulty of maintaining non-free-connex CQs even over insertion-only sequences, we concentrate our attention in this paper to self-join-free free-connex CQs. Our lower bound results are derived under well-known conjectures, including the OMv and OuMv conjectures [15]. We discuss the optimality of our lower bounds, each of which is either an implication or an adaption of Lemma 1.4.

Maintaining CQs over FIFO sequences. We first address the problem of maintaining CQs over FIFO sequences. In this context, we identify a class of queries, termed *weak-q-hierarchical CQs*, which lies strictly between free-connex queries and q-hierarchical queries (see Figure 1). Our first main result is a *structural dichotomy* that determines the hardness of maintaining CQs over FIFO sequences.

Theorem 1.11. *A free-connex CQ Q can be maintained in $O(1)$ time over FIFO sequences if Q is weak-q-hierarchical and cannot be maintained in $O(\sqrt{|D|}^{1-\epsilon})$ time for any $\epsilon > 0$ otherwise, assuming the OMv and OuMv conjectures.*

Maintaining CQs over mixed update sequences. We next consider *mixed* update sequences, in which a specific pattern - either insertion-only or FIFO - is exhibited by a subset of relations. To characterize the complexity of maintaining CQs under such mixed sequences, we introduce a *procedural dichotomy* formalized in the procedureMixDetect (see Algorithm 2). Notably, this result perfectly unifies all the previously known results for arbitrary and insertion-only update sequences together with our new findings for FIFO sequences.

Theorem 1.12. *For a free-connex CQ $Q = (\mathcal{V}, \mathcal{E}, \mathbf{y})$ and a pattern σ of update sequences, Q can be maintained in $O(1)$ time over any σ-sequence if* MixDetect *returns true, and cannot be maintained in $O(\sqrt{|D|}^{1-\epsilon})$ time for any $\epsilon > 0$ otherwise, assuming the OMv and OuMv sconjectures.*

1.4 Organization of This Paper

This paper is organized as follows. In Section 2, we introduce the preliminary concepts essential for understanding both upper and lower bounds. In Section 3, we focus on maintaining CQs over FIFO sequences. We then extend our discussion to maintaining CQs over mixed sequences in Section 4. Finally, we conclude this paper and outline potential directions for future research in Section 5.

2 PRELIMINARIES

2.1 Classification of CQs

We introduce several important classes of CQs that are widely studied in the literature. We start with some commonly used terminology. In a CQ $Q = (\mathcal{V}, \mathcal{E}, \mathbf{y})$, let $\mathcal{E}_x = \{e \in \mathcal{E} : x \in e\}$ be the set of relations containing attribute $x \in \mathcal{V}$. An attribute $x \in \mathcal{V}$ is *unique* if it only appears in one relation, i.e., $|\mathcal{E}_x| = 1$. Let $\mathcal{V}_\bullet$ be the set of *unique* attributes in Q. For a subset $\mathcal{E}' \subseteq \mathcal{E}$ of relations, the $\mathcal{E}'$-induced CQ refers to $Q[\mathcal{E}'] := (\bigcup_{e\in\mathcal{E}'} e, \mathcal{E}', \bigcup_{e\in\mathcal{E}'} (e \cap \mathbf{y}))$.

Acyclic CQ [6, 12, 16]. There are some equivalent definitions of acyclic CQs, and we adopt one based on a generalized join tree as it naturally captures different classes of queries with the height of the tree. A *generalized relation* R_e is defined on a subset $e \subseteq \mathcal{V}$ of attributes such that there exists some input relation $e' \in \mathcal{E}$ with $e \subseteq e'$, and is distinguished from the input relations. A CQ $Q = (\mathcal{V}, \mathcal{E}, \mathbf{y})$ is acyclic if there exists a tree $\mathcal{T}$ in which each node corresponds to a distinct input relation in $\mathcal{E}$ or generalized relation while satisfying the following properties:

- **(cover property)** each input relation corresponds to a node in $\mathcal{T}$; and each leaf node of $\mathcal{T}$ corresponds to an input relation;
- **(connect property)** for each attribute $x \in \mathcal{V}$, all nodes of $\mathcal{T}$ containing x form a connected subtree of $\mathcal{T}$;

$\mathcal{T}$ is called a *generalized join tree* of Q. If all nodes in $\mathcal{T}$ correspond to input relations in $\mathcal{E}$, $\mathcal{T}$ is called a *traditional join tree*.

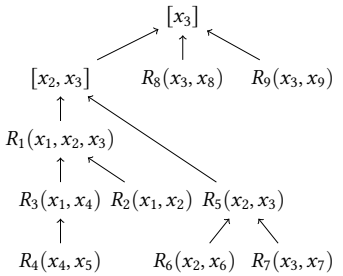


Figure 2. A free-connex join tree for Q_1. $[\cdot]$ is a generalized relation.

Free-connex CQ [5, 25]. An acyclic CQ $Q = (\mathcal{V}, \mathcal{E}, \mathbf{y})$ is *free-connex* if the derived CQ $(\mathcal{V}, \mathcal{E} \cup \{\mathbf{y}\}, \mathbf{y})$ is also acyclic. Equivalently, an acyclic CQ $Q = (\mathcal{V}, \mathcal{E}, \mathbf{y})$ is free-connex if there exists a tree $\mathcal{T}$ in which each node corresponds to a distinct input relation or generalized relation while satisfying the following properties (in addition to the **cover** and **connect property** for acyclic CQs):

- **(above property)** no node corresponding to an input relation is an ancestor of a node corresponding to a generalized relation;
- **(guard property)** if node e corresponds to a generalized relation, $e \subseteq e'$ holds for every child node e' of e;
- **(connex property)** a subset $\mathcal{E}_{\text{con}}$ of nodes exist such that (i) $\mathcal{E}_{\text{con}}$ forms a connected subtree including the root of $\mathcal{T}$; (ii) for any $e \in \mathcal{E}_{\text{con}}$ with its parent e', $e \cap e' \subseteq \mathbf{y}$; (iii) $\mathbf{y} \subseteq \bigcup_{e\in\mathcal{E}_{\text{con}}} e$.

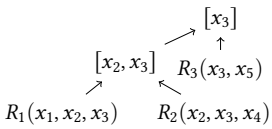


Figure 3. A height-1 free-connex join tree for Q_2. $[\cdot]$ denotes a generalized relation.

$\mathcal{T}$ is called a *free-connex join tree* of Q, and $\mathcal{T}_e$ is the subtree of $\mathcal{T}$ rooted at e. The *height* of $\mathcal{T}$ is the maximum number of relations on any leaf-to-root path, without counting generalized relations.

Example 2.1. Figure 2 shows a free-connex join tree for CQ $Q_1 = R_1(x_1, x_2, x_3) \bowtie R_2(x_1, x_2) \bowtie R_3(x_1, x_4) \bowtie R_4(x_4, x_5) \bowtie R_5(x_2, x_3) \bowtie R_6(x_2, x_6) \bowtie R_7(x_3, x_7) \bowtie R_8(x_3, x_8) \bowtie R_9(x_3, x_9)$. It has height as 3 with the path $[x_3]$ - $[x_2, x_3]$ - R_1 - R_3 - R_4.

q-hierarchical CQ [7]. A CQ $Q = (\mathcal{V}, \mathcal{E}, \mathbf{y})$ is q-hierarchical if for any pair of attributes $x_1, x_2 \in \mathcal{V}$, either $\mathcal{E}_{x_1} \subseteq \mathcal{E}_{x_2}$ or $\mathcal{E}_{x_2} \subseteq \mathcal{E}_{x_1}$ or $\mathcal{E}_{x_1} \cap \mathcal{E}_{x_2} = \emptyset$; and if $x_1 \in \mathbf{y}$ and $\mathcal{E}_{x_1} \subsetneq \mathcal{E}_{x_2}$, then $x_2 \in \mathbf{y}$. We mention two important structural properties of q-hierarchical CQs:

Lemma 2.2 ([25]). *A CQ is* q-hierarchical *if and only if it has a height-1 free-connex join tree.*

Lemma 2.3. *In a q-hierarchical CQ $Q = (\mathcal{V}, \mathcal{E}, \mathbf{y})$ with $\mathcal{E}' \subseteq \mathcal{E}$, $Q[\mathcal{E}']$ is also q-hierarchical.*

Proof of Lemma 2.3. Let $\mathcal{T}$ be a height-1 free-connex join tree of the q-hierarchical CQ Q. For any subset $\mathcal{E}' \subseteq \mathcal{E}$ of relations, we remove all leaf nodes corresponding to relations in $\mathcal{E} - \mathcal{E}'$. We also recursively remove any generalized relation that is a leaf node. The resulting tree is a height-1 free-connex join tree for the $\mathcal{E}'$-induced CQ. Hence, the $\mathcal{E}'$-induced CQ is q-hierarchical. □

Example 2.4. Consider a q-hierarchical CQ $Q_2 := \pi_{x_2,x_3} R_1(x_1, x_2, x_3) \bowtie R_2(x_2, x_3, x_4) \bowtie R_3(x_3, x_5)$ with its height-1 free-connex join tree with $\mathcal{E}_{\text{con}} = \{[x_3], [x_2, x_3]\}$ in Figure 3.

2.2 Upper Bounds

Due to the high time and space costs of the classic change propagation framework [11] for maintaining queries, Wang et al. [25] proposed an improved approach that eliminates the join operator in change propagation [11]. Given a free-connex CQ, the method first organizes each input relation at the leaf level, connecting them via relational operators at all internal nodes. Rather than maintaining subqueries directly, it decomposes each subquery into a semi-join view (V_s) and a projection view (V_p) of linear size. Intuitively, the semi-join view of a relation R contains all tuples in R that can successfully join with tuples in its descendant nodes, while the projection views facilitate the efficient maintenance of these semi-join views. These two views are defined recursively over the join tree. Example 2.5 is used to illustrate these concepts, and additional details can be found in [25].

Example 2.5. Consider a non-q-hierarchical CQ $Q_{\text{hier}} = \pi_{\emptyset} R_1(x_1, x_2) \bowtie R_2(x_1) \bowtie R_3(x_2)$. Figure 4 illustrates the query plan designed for Q_{hier} [25]. In this query plan, each node e is associated with a semi-join view $V_s(R_e)$ that stores those tuples in R_e that can be successfully joined with all relations in the subtree rooted at e. For instance, we define the semi-join views as follows: $V_s(R_1) := R_1 \ltimes R_2$, $V_s(R_2) := R_2$ and $V_s(R_3) := R_3 \ltimes (R_1 \ltimes R_2)$. Each node e is further associated with a projection view $V_p(R_e)$ that stores the projection of the corresponding semi-join view onto the join attributes between e and its parent node. For instance, we define the projection-views as follows: $V_p(R_1) := \pi_{x_2} V_s(R_1)$, $V_p(R_2) := \pi_{x_1} V_s(R_2)$ and $V_s(R_3) := \pi_{\emptyset} V_p(R_3)$. It is noteworthy that the series of semi-joins within the subtree rooted at any node e can be compactly represented by the semi-joins between the node e and the projection views of its child nodes. Consequently, the simplified expressions for the semi-join views become: $V_s(R_1) := R_1 \ltimes V_p(R_2)$ and $V_s(R_3) := R_3 \ltimes V_p(R_1)$.

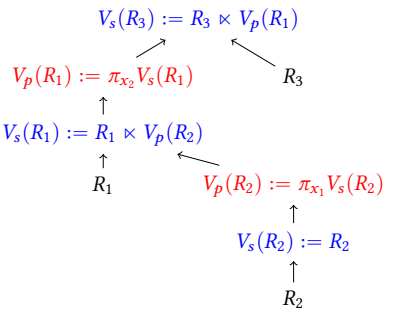


Figure 4. Maintaining Q_{hier} by the framework proposed in [25]. Input relations are in black, semi-join views are in blue, and projection views are in red.

This method guarantees both linear space usage and maintenance time in the worst case; however, the worst-case complexity may not manifest for every update sequence. To provide a more nuanced

cost analysis, [25] introduced the notion of *enclosureness* for update sequences to quantify the amortized update time required. Intuitively, the "status" of a tuple t, i.e., whether t is present in $V_s(R_e)$, can change multiple times during its lifespan as a result of insertions or deletions of tuples in descendant nodes. Notably, successive insertions (when t is present in $V_s(R_e)$) or successive deletions (when t is absent from $V_s(R_e)$) in descendant nodes do not further change its status. For example, if an insertion of a tuple t' in the subtree rooted at e triggers a change in the status of t, then t will remain in the semi-join view $V_s(R_e)$ until the first subsequent deletion in the subtree occurs. Consequently, the concept of a tuple's *minimal lifespan* is introduced to capture the minimum temporal interval over which a tuple can effect a change in the status of another tuple.

Definition 2.6 (Minimal lifespan). Given a free-connex query Q, a free-connex $\mathcal{T}$ of Q, an update sequence S, and a tuple t_1 in an input relation R_e, the two *minimal lifespans* of t_1 with respect to its lifespan $I(t_1) = [t_1^+, t_1^-]$ are defined as:

$$\widehat{I}(t_1) = \left[t_1^+, \min\left(t_1^-, \min_{t_2 \in R_{e'} : e' \in \mathcal{T}_e - \{e\}, t_2^- > t_1^+} t_2^- \right) \right];$$

$$\check{I}(t_1) = \left[\max\left(t_1^+, \max_{t_2 \in R_{e'} : e' \in \mathcal{T}_e - \{e\}, t_2^+ < t_1^-} t_2^+ \right), t_1^- \right]$$

Definition 2.7 (Enclosureness). Given a free-connex CQ Q, a free-connex join tree $\mathcal{T}$ of Q, and an update sequence S, for a node $e \in \mathcal{T}$ and a tuple $t \in R_e$, its *enclosureness* is

$$\lambda_{\mathcal{T}}(t) = \max_{\substack{\text{For all } t' \in \mathcal{J}, \text{ there exists } e' \in \mathcal{T}_e - \{e\}, t' \in R_{e'} \\ \text{For all } t_1 \in \mathcal{J}, \dot{I}(t_1) \subseteq I(t) \\ \text{For all } t_2, t_3 \in \mathcal{J}, \dot{I}(t_2) \cap \dot{I}(t_3) = \varnothing}} |\mathcal{J}|,$$

where each $\dot{I}$ is either $\widehat{I}$ or $\check{I}$, i.e., the largest number of disjoint minimal lifespans of tuples in the descendants of e, which are contained in the lifespan of t. Then, the enclosureness of the update sequence is still average: $\lambda_{\mathcal{T}}(S) := \max\left(\frac{\sum_{t \in S} \lambda_{\mathcal{T}}(t)}{|S|}, 1 \right)$.

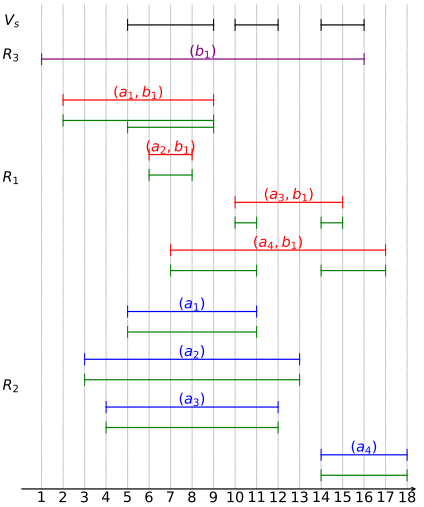


Figure 5. An update sequence for Q_{hier}. Red intervals are lifespans of tuples in R_1. Blue intervals are lifespans of tuples in R_2. Purple intervals are lifespans of tuples in R_3. Green intervals are minimal lifespans for tuples in R_1 and R_2. Black intervals are the insertions and deletions of $(b_1) \in V_s(R_3)$.

This concept enables a more refined analysis of update sequences that exhibit varying patterns. For instance, in an insertion-only sequence, the enclosureness is 1 on a free-connex join tree. Consequently, any free-connex conjunctive query can be maintained in $O(1)$ time over insertion-only update sequences.

Example 2.8. Consider the example of maintaining $Q_{\text{hier}} = \pi_{\varnothing} R_1(x_1, x_2) \bowtie R_2(x_1) \bowtie R_3(x_2)$ under an update sequence as shown in Figure 5. The relevant views to be maintained are shown in Figure 4. Observe that tuple (b_1) is inserted into R_3 at timestamp 1 and deleted from R_3 at timestamp 16. It is first inserted into $V_s(R_3)$ at timestamp 5 due to the insertion of tuple $(a_1) \in R_2$, and subsequently deleted from $V_s(R_3)$ at timestamp 9 following the deletion of tuple $(a_1, b_1) \in R_1$. Consequently, the tuple $(a_1) \in R_2$ has a minimal lifespan as $[5, 9]$. Moreover, although tuple (b_1) can also join with tuple $(a_2) \in R_2$ and tuple $(a_1, b_1) \in R_1$ during the period $[6, 8]$, the insertion of tuple (a_2, b_1) won't change the status of (b_1) in $V_s(R_3)$, as the successive insertion or deletion do not trigger additional changes to its status. Subsequently, tuple (b_1) is reinserted into $V_s(R_3)$ at timestamp 10 and removed at timestamp 12. During this phase, the tuple $(a_3, b_1) \in R_1$ has a minimal

Algorithm 1: TRIM($Q = (\mathcal{V}, \mathcal{E}, \mathbf{y})$)

$\mathcal{V}_\bullet \leftarrow \{x \in \mathcal{V} : |\mathcal{E}_x| = 1\}$;
while true do
  if *there exists* $e, e' \in \mathcal{E}$ *s.t.* $e - \mathcal{V}_\bullet \subseteq e'$ *and* $e \cap \mathcal{V}_\bullet \subseteq \bar{\mathbf{y}}$ **then** $\mathcal{E} \leftarrow \mathcal{E} - \{e\}$;
  if *there exists* $e, e' \in \mathcal{E}$ *s.t.* $e - \mathcal{V}_\bullet \subseteq e' \cap \mathbf{y}$ **then** $\mathcal{E} \leftarrow \mathcal{E} - \{e\}$;
  if $\mathcal{E}$ *does not change* **then break**;
return $\mathcal{E}$;

lifespan as $[10, 11]$, as it is inserted at timestamp 10 and its status is changed by the deletion of tuple $(a_1) \in R_2$ at timestamp 11. Finally, tuple (b_1) is inserted into $V_s(R_3)$ once more at timestamp 14 due to the insertion of $(a_4) \in R_2$, and it is ultimately removed from $V_s(R_3)$ at timestamp 16 as a result of the deletion of (b_1) itself. Taken together, the enclosureness of (b_1) is 3, with one possible disjointed set of minimal lifespans being $\{[5, 9], [10, 11], [14, 15]\}$.

3 MAINTAIN CQS OVER FIFO SEQUENCES

3.1 Weak-q-hierarchical CQs

We begin by introducing two key concepts for CQs: *reducible relations* and *skeleton*. These notions play a crucial role in characterizing the class of weak-q-hierarchical CQs. In particular, by understanding how relations within a query can be reduced relative to one another and how the remaining non-reducible relations form a skeleton, we can gain further insight into the properties and efficient maintenance of these queries.

Definition 3.1 (Reducible Relation). In a free-connex CQ $Q = (\mathcal{V}, \mathcal{E}, \mathbf{y})$, a relation $e \in \mathcal{E}$ is reducible if there exists another relation $e' \in \mathcal{E}$ such that (i) $e - \mathcal{V}_\bullet \subseteq e'$ and $e \cap \mathcal{V}_\bullet \subseteq \bar{\mathbf{y}}$; or (ii) $e - \mathcal{V}_\bullet \subseteq e' \cap \mathbf{y}$. In this context, e' is called an *anchor* of e.

It is worth noting that one relation could have multiple anchors, and one relation could serve as an anchor for multiple other relations. Moreover, the anchor ordering is transitive. Specifically, if e' is an anchor of e, and e'' is an anchor of e', then e'' is an anchor of e.

LEMMA 3.2. *If e_1 is reducible by e_2, and e_2 is reducible by e_3, then e_1 is reducible by e_3.*

PROOF. We distinguish the following four cases. **Case 1:** $e_1 - \mathcal{V}_\bullet \subseteq e_2$, $e_1 \cap \mathcal{V}_\bullet \subseteq \bar{\mathbf{y}}$, and $e_2 - \mathcal{V}_\bullet \subseteq e_3$, $e_2 \cap \mathcal{V}_\bullet \subseteq \bar{\mathbf{y}}$: from $e_1 - \mathcal{V}_\bullet \subseteq e_2$, $e_2 - \mathcal{V}_\bullet \subseteq e_3$, and $e_1 \cap (e_2 \cap \mathcal{V}_\bullet) = \emptyset$, we have $e_1 - \mathcal{V}_\bullet \subseteq e_2 - \mathcal{V}_\bullet \subseteq e_3$, therefore e_1 and e_3 satisfies the condition (i) in Definition 3.1. **Case 2:** $e_1 - \mathcal{V}_\bullet \subseteq e_2$, $e_1 \cap \mathcal{V}_\bullet \subseteq \bar{\mathbf{y}}$, and $e_2 - \mathcal{V}_\bullet \subseteq e_3 \cap \mathbf{y}$: similarly, from $e_1 - \mathcal{V}_\bullet \subseteq e_2$ and $e_2 - \mathcal{V}_\bullet \subseteq e_3 \cap \mathbf{y}$, we can get $e_1 - \mathcal{V}_\bullet \subseteq e_2 - \mathcal{V}_\bullet \subseteq e_3 \cap \mathbf{y}$, therefore e_1 and e_3 satisfies the condition (ii) in Definition 3.1. **Case 3:** $e_1 - \mathcal{V}_\bullet \subseteq e_2 \cap \mathbf{y}$, and $e_2 - \mathcal{V}_\bullet \subseteq e_3$, $e_2 \cap \mathcal{V}_\bullet \subseteq \bar{\mathbf{y}}$: since $e_2 \cap \mathcal{V}_\bullet \subseteq \bar{\mathbf{y}}$, $e_2 \cap \mathbf{y} \subseteq e_2 - \mathcal{V}_\bullet$. Therefore, $e_1 - \mathcal{V}_\bullet \subseteq e_2 \cap \mathbf{y} \subseteq e_2 - \mathcal{V}_\bullet \subseteq e_3$, which leads to $e_1 - \mathcal{V}_\bullet \subseteq e_3 \cap \mathbf{y}$ that satisfies the condition (ii) in Definition 3.1. **Case 4:** $e_1 - \mathcal{V}_\bullet \subseteq e_2 \cap \mathbf{y}$, and $e_2 - \mathcal{V}_\bullet \subseteq e_3 \cap \mathbf{y}$: similarly, $e_1 - \mathcal{V}_\bullet \subseteq e_2 \cap \mathbf{y} \subseteq e_2 - \mathcal{V}_\bullet \subseteq e_3 \cap \mathbf{y}$, making e_1 and e_3 satisfies condition (ii) in Definition 3.1. □

We next introduce the TRIM procedure, as described in Algorithm 1, which takes as input a CQ $Q = (\mathcal{V}, \mathcal{E}, \mathbf{y})$ and returns a subset of relations, referred to as the *skeleton* of Q. This procedure iteratively identifies a *reducible relation* and its *anchor* in the remaining relations (if possible), and removes this reducible relation. When no more relations can be removed, the remaining subset of relations is returned. The skeleton returned (denoted as $\mathcal{E}_\clubsuit$) may not be unique. Below, we use $Q_\clubsuit$ to denote the CQ induced by $\mathcal{E}_\clubsuit$.

Example 3.3. Consider a CQ $Q_5 = R_1(x_1, x_2) \bowtie R_2(x_1, x_2, x_3) \bowtie R_3(x_2, x_3, x_4) \bowtie R_4(x_2, x_3, x_5) \bowtie R_5(x_4, x_6)$. We have $\mathcal{V}_\bullet = \{x_5, x_6\}$. A skeleton is $\{e_2, e_3\}$, since $e_1 \subsetneq e_2$, $e_4 - \mathcal{V}_\bullet \subsetneq e_2$, and $e_5 - \mathcal{V}_\bullet \subsetneq e_3$. On the other hand, consider another CQ $Q_6 = R_1(x_1, x_2) \bowtie R_2(x_2, x_3) \bowtie R_3(x_2, x_4)$. We have $\mathcal{V}_\bullet = \{x_1, x_3, x_4\}$. One possible skeleton is $\{e_1, e_2\}$ and the other possible skeleton is $\{e_1, e_3\}$.

Definition 3.4 (weak-q-hierarchical). A free-connex CQ Q is weak-q-hierarchical if it has a skeleton that induces a q-hierarchical CQ.

Example 3.5. For a weak-q-hierarchical CQ $Q_3 = \pi_{x_2} R_1(x_1, x_2) \bowtie R_2(x_2, x_4) \bowtie R_3(x_2, x_5) \bowtie R_4(x_1) \bowtie R_5(x_3, x_6)$, Algorithm 1 removes R_4, which is reducible with R_1 as its anchor and R_5, which is reducible with R_2 as its anchor. Remaining relations R_1, R_2, R_3, form the skeleton and induce a q-hierarchical CQ $\pi_{x_2} R_1(x_1, x_2) \bowtie R_2(x_2, x_4) \bowtie R_3(x_2, x_5)$.

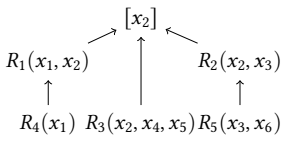


Figure 6. A height-2 free-connex join tree for Q_3.

3.2 Maintaining Weak-q-hierarchical CQs

Next, we demonstrate that weak-q-hierarchical CQs can be efficiently maintained over FIFO update sequences. Before introducing the algorithm, we establish two key structural properties of weak-q-hierarchical CQs with respect to their free-connex join trees.

Lemma 3.6. *For a free-connex CQ $Q = (\mathcal{V}, \mathcal{E}, \mathbf{y})$ with its free-connex join tree $\mathcal{T}$, for any leaf node e with its parent node e', e is reducible, with e' serving as its anchor.*

Proof of Lemma 3.6. The connect property implies $e - \mathcal{V}_\bullet = e \cap e'$. If $e \cap \mathcal{V}_\bullet = \emptyset$, condition (i) follows. Below, assume $e \cap \mathcal{V}_\bullet \neq \emptyset$. If $(e \cap \mathcal{V}_\bullet) \cap \mathbf{y} = \emptyset$, condition (i) follows. If $(e \cap \mathcal{V}_\bullet) \cap \mathbf{y} \neq \emptyset$, we must have $e \in \mathcal{E}_{\text{con}}$; otherwise, no relation in $\mathcal{E}_{\text{con}}$ contains the output attribute(s) in $(e \cap \mathcal{V}_\bullet) \cap \mathbf{y}$. As $e \in \mathcal{E}_{\text{con}}$, we must have $e' \in \mathcal{E}_{\text{con}}$; otherwise, $\mathcal{E}_{\text{con}}$ does not form a connected subtree including the root of $\mathcal{T}$. Implied by the property of $\mathcal{E}_{\text{con}}$, $e \cap e' \subseteq \mathbf{y}$, and condition (ii) follows. □

From Lemma 3.6, when constructing a free-connex join tree, a reducible relation can always be placed as a leaf node and thus, a child node of any of its anchors.

Lemma 3.7. *A CQ is weak-q-hierarchical if and only if it has a free-connex join tree of height $\leqslant 2$.*

Proof. *Only-If Direction.* Let Q be a weak-q-hierarchical CQ. As $Q_\clubsuit$ is q-hierarchical, it has a height-1 free-connex join tree $\mathcal{T}_\clubsuit$, with a one-to-one correspondence between leaf nodes in $\mathcal{T}_\clubsuit$ and relations in $\mathcal{E}_\clubsuit$. For each relation $e \in \mathcal{E} - \mathcal{E}_\clubsuit$, it is always feasible to find some $e' \in \mathcal{E}_\clubsuit$ such that one of the two conditions of reducible relations is satisfied by the transitive property of anchor. We add e as a child of e'. The resulting tree $\mathcal{T}$ is a height-2 free-connex join tree for Q.

If Direction. Let $\mathcal{T}$ be a free-connex join tree of Q. If $\mathcal{T}$ has a height of 1, Q is q-hierarchical by Lemma 2.2. Implied by Lemma 2.3, the $\mathcal{E}_\clubsuit$-induced CQ is q-hierarchical. We next focus on the case when $\mathcal{T}$ has a height of 2, with the connex subset $\mathcal{E}_{\text{con}}$. Let $\mathcal{V}_\bullet$ be the set of unique attributes in Q. Implied by Lemma 3.6, each leaf node e is reducible with its parent node e' as the anchor. Let $\mathcal{T}'$ be the residual tree by removing all leaf nodes of $\mathcal{T}$. Let $\mathcal{E}' \subseteq \mathcal{E}$ be the set of input relations in $\mathcal{T}'$. As $\mathcal{T}$ has height 2, $\mathcal{T}'$ has height 1. From Lemma 2.2, $Q[\mathcal{E}']$ is q-hierarchical. Let $\mathcal{E}_\clubsuit$ be the skeleton of $\mathcal{E}'$ by invoking Trim to $Q[\mathcal{E}']$. As every leaf node $e \in \mathcal{T}$ is reducible with its parent as an anchor, $\mathcal{E}_\clubsuit$ is also a skeleton of $\mathcal{E}$. From Lemma 2.3, $Q_\clubsuit$ is also q-hierarchical, since $\mathcal{E}_\clubsuit \subseteq \mathcal{E}'$. □

We highlight a critical observation regarding the enclosureness of FIFO update sequences in Lemma 3.8. Combining it with Lemma 3.7, we prove the first half of Theorem 1.11.

Lemma 3.8 ([25], Lemma 6.8). *For a free-connex CQ Q with a height-2 free-connex join tree $\mathcal{T}$, every FIFO sequence for Q has enclosureness as 1 under $\mathcal{T}$.*

3.3 Hardness of Maintaining Non-Weak-q-hierarchical CQs

To demonstrate the difficulty of maintaining non-weak-q-hierarchical CQs, we begin by considering two small queries that fall into this class:

$$Q_{\text{hook}} = \pi_{x_1}\left(R_1(x_1, x_2) \bowtie R_2(x_2, x_3) \bowtie R_3(x_3)\right)$$
$$Q_{\text{path}} = \pi_{\varnothing}\left(R_1(x_1) \bowtie R_2(x_1, x_2) \bowtie R_3(x_2, x_3) \bowtie R_4(x_3, x_4) \bowtie R_5(x_4)\right)$$

We prove the hardness for maintaining Q_{hook} over FIFO sequences in Theorem 3.9 and Q_{path} in Theorem 3.10 separately.

Theorem 3.9. *No index for Q_{hook} can be updated in $O(\sqrt{|D|}^{1-\epsilon})$ amortized time over FIFO sequences while supporting $O(\sqrt{|D|}^{1-\epsilon})$-delay enumeration for any $\epsilon > 0$, assuming OMv conjecture.*

Proof. Given an arbitrary OMv-instance, we encode the matrix M by $R_1(x_1, x_2)$ and the vectors $\langle v_i : i \in [n]\rangle$ by a subquery $\pi_{x_2}\left(R_2(x_2, x_3) \bowtie R_3(x_3)\right)$. See Figure 7. We construct an update sequence S for Q_{hook} as follows: (1) we first add a tuple $t = (i, j)$ into R_1 with lifespan $[n^2 + in - j, 3n^2 + in - j]$ if $M_{ij} \neq 0$; (2) we add a tuple $t = (i)$ with lifespan $[in, in + 2n^2]$ into R_3, for each $i \in [n]$; (3) after receiving vector v_i, we add a tuple $t = (j, i)$ into R_2 with lifespan $[2n^2 + in - j, 4n^2 + in - j]$ if $(v_i)_j \neq 0$; (4) we issue the enumeration query at time $in + 2n^2$: for each result j enumerated for Q_{hook}, we set $(Mv_i)_j = 1$; (5) we repeat (3)-(4) for the next v_{i+1} until n vectors are processed. Each tuple has the same lifespan; hence, S is a FIFO sequence. If an index can be updated in $O(n^{1-\epsilon})$ time while supporting $O(n^{1-\epsilon})$-delay enumeration for Q_{hook} over FIFO sequences, the OMv problem can be solved in $O(n^2 \cdot n^{1-\epsilon} + n^2 \cdot n^{1-\epsilon}) = O(n^{3-\epsilon})$ time. As $|D| = O(n^2)$, we have $n \geqslant \sqrt{|D|}$.

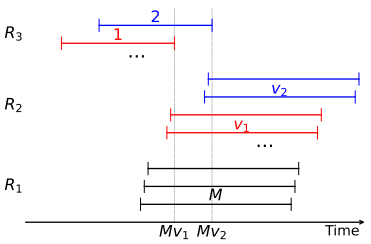


Figure 7. A reduction from an OMv instance to a σ-sequence for Q_{hook}, where $\sigma(\{e\})$ is FIFO for every e.

Theorem 3.10. *No index for Q_{path} can be updated in $O(\sqrt{|D|}^{1-\epsilon})$ time over FIFO sequences while supporting $O(\sqrt{|D|}^{2-\epsilon})$-delay enumeration for any $\epsilon > 0$, assuming the OuMv conjecture.*

Proof. Given an arbitrary OuMv-instance, we encode the matrix M by $R_3(x_2, x_3)$ and the pairs of vectors $\langle (u_i, v_i) : i \in [n]\rangle$ by two subqueries $\pi_{x_2}\left(R_2(x_2, x_1) \bowtie R_1(x_1)\right)$ and $\pi_{x_3}\left(R_4(x_3, x_4) \bowtie R_5(x_4)\right)$. See Figure 8. We construct an update sequence S for Q_{path} as follows: (1) add a tuple $t = (i, j)$ into R_3 with lifespan $[2n^2 + in - j, 6n^2 + in - j]$, for each pair $(i, j) \in [n] \times [n]$ if $M_{ij} \neq 0$; (2) add a tuple $t = (i)$ with lifespan $[(2i - 1)n, (2i - 1)n + 4n^2]$ into R_1 and $[2in, 2in + 4n^2]$ into R_5; (3) for each pair of vectors (u_i, v_i), add a tuple $t = (i, j)$ into R_2 with lifespan $[(2i - 1)n + 4n^2 - j, (2i - 1)n + 8n^2 - j]$ if $(u_i)_j \neq 0$, and similarly add a tuple $t = (j, i)$ into R_4 with lifespan $[2in + 4n^2 - j, 2in + 8n^2 - j]$ if $(v_i)_j \neq 0$; (4) issue the enumeration query at time $2in + 4n^2$: if the result is true for Q_{path}, we output true for $u_i^T M v_i$, and false otherwise; (5) repeat (3)-(4) for the next pair (u_{i+1}, v_{i+1}), until n pairs of vectors are processed. Each tuple has the same lifespan; hence S is a FIFO sequence. If an index for Q_{path} can be updated in $O(n^{1-\epsilon})$ time over FIFO sequences while supporting $O(n^{2-\epsilon})$-delay enumeration, OuMv can be solved in $O(n^2 \cdot n^{1-\epsilon} + n \cdot n^{2-\epsilon}) = O(n^{3-\epsilon})$ time. As $|D| = O(n^2)$, we have $n \geqslant \sqrt{|D|}$.

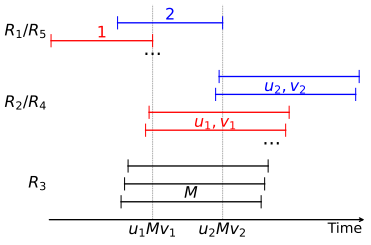


Figure 8. A reduction from an OuMv instance to a σ-sequence for Q_{path}, where $\sigma(\{e\})$ is FIFO for every e.

Algorithm 2: MixDetect$(Q = (\mathcal{V}, \mathcal{E}, \mathbf{y}), \sigma)$

1 **if** $\exists$ *a reducible relation* e *with its anchor* e' *s.t.* $\sigma(\{e\})$ *and* $\sigma(\{e'\})$ *are insertion-only* **then**
2 **return** MixDetect$(Q[\mathcal{E} - \{e\}], \pi_{\mathcal{E}-\{e\}}\sigma)$;
3 **else if** Q *is non-weak-q-hierarchical* **then return false**;
4 **else if** Q *does not have a free-connex join tree compatible with* σ **then return false**;
5 **return true**;

We can extend the hardness result to general free-connex but non-weak-q-hierarchical CQs, as established in Theorem 3.12. The proof leverages two key structures identified by Lemma 3.11. All omitted proofs are given in Appendix A.

Lemma 3.11. *For any free-connex but non-weak-q-hierarchical CQ* $Q = (\mathcal{V}, \mathcal{E}, \mathbf{y})$*, it must contain at least one of the following structures:*

- **(hook core)** *three attributes* $x_1 \in \mathbf{y}, x_2, x_3 \in \bar{\mathbf{y}}$ *and three relations* $e_1, e_2, e_3 \in \mathcal{E}$ *such that* $x_1 \in e_1, x_2 \in e_1 \cap e_2 - e_3, x_3 \in e_2 \cap e_3$*, and no relation* $e \in \mathcal{E}$ *contains both* x_1, x_3*.*
- **(path core)** *four attributes* $x_1, x_2, x_3, x_4 \in \mathcal{V}$ *and five relations* $e_1, e_2, e_3, e_4, e_5 \in \mathcal{E}$ *such that* $x_1 \in e_1 \cap e_2, x_2 \in e_2 \cap e_3, x_3 \in e_3 \cap e_4, x_4 \in e_4 \cap e_5$*, and no relation* $e \in \mathcal{E}$ *contains* x_i, x_j *with* $|j - i| > 1$*.*

Theorem 3.12. *For any free-connex but non-weak-q-hierarchical CQ and any* $\epsilon > 0$*, no index can be updated in* $O(\sqrt{|D|}^{1-\epsilon})$ *amortized time over FIFO sequences while supporting* $O(\sqrt{|D|}^{1-\epsilon})$*-delay enumeration, assuming the OuMv and OMv conjectures.*

4 MAINTAIN CQS OVER MIXED SEQUENCES

This section presents a procedure, MixDetect, which determines the hardness of maintaining a CQ Q over σ-sequences, as described in Algorithm 2. Note that when MixDetect(Q, σ) returns true, Q can be maintained in $O(1)$ amortized time over σ-sequences; otherwise, it cannot be maintained in $O(\sqrt{N}^{1-\epsilon})$ amortized time for any $\epsilon > 0$. All missing proofs are given in Appendix B.

As described in Algorithm 2, the high-level idea is to iteratively simplify the input CQ and update sequences until a base case is reached. First, a reducible relation e is removed if it has an anchor e' such that both $\sigma(\{e\})$ and $\sigma(\{e'\})$ are insertion-only (lines 1-2, Lemma 4.3). Next, the procedure returns **false** if Q is not weak-q-hierarchical (line 3, Lemma 4.4) or does not have a free-connex join tree *compatible* with σ (line 4, Lemma 4.8). If neither condition is met, the procedure returns **true** (line 4, Lemma 4.16). Although different orderings may be used for removing reducible relations, they always yield the same final result.

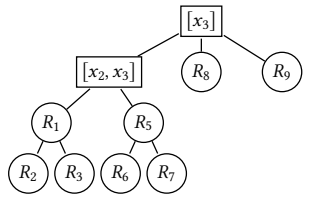


Figure 9. A compatible free-connex join tree for Q_4.

Definition 4.1 (Compatible Free-Connex Join Tree). For a weak-q-hierarchical CQ $Q = (\mathcal{V}, \mathcal{E}, \mathbf{y})$ and a pattern σ of update sequences, a free-connex join tree $\mathcal{T}$ for Q is compatible with σ if for each leaf node e in $\mathcal{T}$ with its parent node e' (where both e and e' correspond to input relations of $\mathcal{E}$), the following conditions hold: (1) e' is an anchor of e; and (2) either $\sigma(\{e\})$ is insertion-only, or $\sigma(\{e, e'\})$ is FIFO.

Example 4.2. Consider Example 2.1 with a pattern σ of update sequences where $\sigma(\{e_1\})$, $\sigma(\{e_8\})$ and $\sigma(\{e_9\})$ are arbitrary, $\sigma(\{e_2, e_3, e_4\})$ is insertion-only and $\sigma(\{e_5, e_6, e_7\})$ is FIFO. We remove e_4 since e_4 is reducible with e_3 serving as an anchor, and $\sigma(\{e_3\}), \sigma(\{e_4\})$ are insertion-only. The resulting CQ $Q_4 = R_1(x_1, x_2, x_3) \bowtie R_2(x_1, x_2) \bowtie R_3(x_1, x_4) \bowtie R_5(x_2, x_3) \bowtie R_6(x_2, x_6) \bowtie R_7(x_3, x_7) \bowtie$

$R_8(x_3, x_8) \bowtie R_9(x_3, x_9)$ is weak-q-hierarchical with a compatible free-connex join tree, as illustrated in Figure 9. Consequently, Algorithm 2 returns **true**.

Let's continue with Example 4.2 by considering a different pattern σ of update sequences, with the only exception being that $\sigma(\{e_4\})$ is arbitrary. In this case, the simplification step is not applied, and the resulting query is not weak-q-hierarchical. As a result, Algorithm 2 returns **false**.

The remainder of this section is organized as follows. In Section 4.1, we prove the correctness of lines 1-2 in Algorithm 2, i.e., the simplification step does not change the problem's hardness. We then prove the correctness of three base cases of Algorithm 2: line 3 in Section 4.2, line 4 in Section 4.3 and line 5 in Section 4.4.

4.1 Simplification Preserves Hardness: Lines 1-2 of Algorithm 2

Lemma 4.3. *For a free-connex CQ $Q = (\mathcal{V}, \mathcal{E}, \mathbf{y})$ and a pattern σ of update sequences, for any reducible relation e and its anchor e', if both $\sigma(\{e\})$ and $\sigma(\{e'\})$ are insertion-only, then Q can be maintained in $O(1)$ time over σ-sequences if and only if $Q[\mathcal{E} - \{e\}]$ can be maintained in $O(1)$ time over $\pi_{\mathcal{E}-\{e\}}\sigma$-sequences.*

Proof. We give a running example in Appendix B to help illustrate the algorithms used in the proof below. For simplicity, let $Q' = Q[\mathcal{E} - \{e\}]$.

If Direction. Let $\mathcal{A}'$ be an algorithm maintaining Q' in $O(1)$ time over $\pi_{\mathcal{E}-\{e\}}\sigma$-sequences. For a σ-sequence S for Q, we construct a sequence S' for Q' as follows. For every update $u = (t, s, +, R_{e''}) \in S$, if $e'' \in \mathcal{E} - \{e, e'\}$, we add it to S'. We maintain the results of $R_{e'} \ltimes R_e$ on the update sequence $\pi_{e,e'}S$. For every delta result $t \in R_{e'} \ltimes R_e$ that is generated at timestamp s, we insert $u = (t, s, +, R_{e'})$ into the S'. As all updates to $R_{e'}$ are insertion-only, S' is a $\pi_{\mathcal{E}-\{e\}}\sigma$-sequence. We run algorithm $\mathcal{A}'$ for Q' over S'. For each result t enumerated by $\mathcal{A}'$, we enumerate all results in $\pi_{\mathbf{y}}(R_e \bowtie t)$ for Q within $O(1)$ delay if $e \cap \mathbf{y} \cap \mathcal{V}_\bullet \neq \emptyset$, and enumerate t directly otherwise. As both $\pi_e S$ and $\pi_{e'} S$ are insertion-only, the delta of $R_{e'} \ltimes R_e$ can be maintained in $O(1)$ amortized time, and enumerated within $O(1)$ delay. Hence, Q can be maintained in $O(1)$ time over σ-sequences.

Only-If Direction. Let $\mathcal{A}$ be an algorithm that can maintain Q in $O(1)$ time over any σ-sequence. We next show another algorithm $\mathcal{A}'$ for Q'. Given a $\pi_{\mathcal{E}-\{e\}}\sigma$-sequence S' for Q', we construct a sequence S for Q as follows. For every update $u \in S'$, we add it to S. For every update $u = (t, s, +, R_{e'}) \in S'$, we add an update $u' = (t', s, +, R_e)$ to S, such that $\pi_{e\cap e'} t' = \pi_{e\cap e'} t$ and $\pi_x t' = \{*\}$ for every attribute $x \in e - e'$. All updates to R_e are insertion-only, hence, S is a σ-sequence. We run algorithm $\mathcal{A}$ for Q over S. For every result t enumerated for Q by $\mathcal{A}$, we output $t' = \pi_{\mathbf{y}-e\cap\mathcal{V}_\bullet} t$ for Q'. As $\pi_{e\cap\mathcal{V}_\bullet} t = \{*\}$ (if $e \cap \mathcal{V}_\bullet \neq \emptyset$) for every tuple t enumerated for Q, and there is no pair of identical tuples enumerated for Q, all tuples enumerated for Q' are also distinct. Putting everything together, $\mathcal{A}'$ can maintain Q' in $O(1)$ time over any $\pi_{\mathcal{E}-\{e\}}\sigma$-sequence. □

4.2 Base Case 1: Line 3 of Algorithm 2

Lemma 4.4. *For a free-connex CQ and a pattern σ of update sequences, to which no simplification step can be applied, if Q is non-weak-q-hierarchical, Q cannot be maintained in $O\left(\sqrt{|D|}^{1-\epsilon}\right)$ time over σ-sequences, assuming the OuMv and OMv conjectures.*

Let's start by revisiting two simplest free-connex but non-weak-q-hierarchical CQs, Q_{hook} and Q_{path}. In Theorems 3.9 and 3.10, we already have demonstrated the difficulty of maintaining Q_{hook} and Q_{path} over σ-sequences when $\sigma(\mathcal{E})$ is FIFO. Recall that $\mathcal{E}$ is the set of all relations. The condition that $\sigma(\mathcal{E})$ is FIFO also implies that $\sigma(\mathcal{E}')$ is FIFO for an arbitrary subset $\mathcal{E}' \subseteq \mathcal{E}$ of relations. We now extend this hardness condition in Examples 4.5 and 4.6.

Example 4.5. For $Q_{\text{hook}} = \pi_{x_1} R_1(x_1, x_2) \bowtie R_2(x_2, x_3) \bowtie R_3(x_3)$, and a pattern σ of update sequences where either $\sigma(\{e_3\})$ or $\sigma(\{e_2\})$ is not insertion-only, no index can be updated in $O(\sqrt{|D|}^{1-\epsilon})$ amortized time over σ-sequences while supporting $O(\sqrt{|D|}^{1-\epsilon})$-delay enumeration for any $\epsilon > 0$, assuming the OMv conjecture [7]. Given an OMv instance, we use R_1 to encode the matrix and the following subquery $R_4 = \pi_{x_2} R_2(x_2, x_3) \bowtie R_3(x_3)$ to encode vectors. We essentially maintain $Q_{\text{qcore}} = \pi_{x_1} R_1(x_1, x_2) \bowtie R_4(x_2)$ over σ'-sequences, where $\sigma'(\{e_4\})$ is arbitrary. The hardness result follows [7]. See Figures 10 - 11 for illustrations.

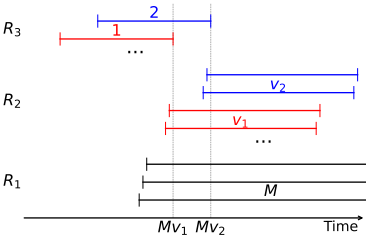


Figure 10. A reduction from an OMv instance to a σ-sequence for Q_{hook}, where $\sigma(\{e_3\})$ is not insertion-only.

Example 4.6. For $Q_{\text{path}} = \pi_{\varnothing} R_1(x_1) \bowtie R_2(x_1, x_2) \bowtie R_3(x_2, x_3) \bowtie R_4(x_3, x_4) \bowtie R_5(x_4)$, and a pattern σ of update sequences such that either $\sigma(\{e_1\})$ or $\sigma(\{e_2\})$ is not insertion-only, and either $\sigma(\{e_4\})$ or $\sigma(\{e_5\})$ is not insertion-only, no index can be updated in $O(\sqrt{|D|}^{1-\epsilon})$ amortized time over σ-sequences while supporting $O(\sqrt{|D|}^{2-\epsilon})$-delay enumeration for any $\epsilon > 0$, assuming the OuMv conjecture. Given an instance of OuMv, we use R_3 to encode the matrix, and use the following two subqueries $R_6 = \pi_{x_2} R_1(x_1) \bowtie R_2(x_1, x_2)$, $R_7 = \pi_{x_3} R_4(x_3, x_4) \bowtie R_5(x_4)$ to encode pairs of vectors separately. We essentially maintain $Q_{\text{hier}} = \pi_{\varnothing}(R_6(x_2) \bowtie R_3(x_2, x_3) \bowtie R_7(x_3))$ over σ'-sequences, where $\sigma'(\{e_6\})$ and $\sigma'(\{e_7\})$ are arbitrary. The hardness result follows [7]. For an illustration, see Figure 12 which handles the case when $\sigma(e_1)$ and $\sigma(e_4)$ are not insertion-only. The remaining cases can be proved similarly; additional details are given in Appendix B.

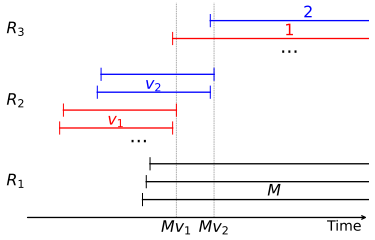


Figure 11. A reduction from an OMv instance to a σ-sequence for Q_{hook}, where $\sigma(\{e_2\})$ is not insertion-only.

We can now extend these hardness results to general free-connex but non-weak-q-hierarchical CQs falling in this case, by reducing either Q_{hook} or Q_{path} to the given query with the help of Lemma 3.11. See an example below:

Example 4.7. Let's continue with the CQ Q_1 as illustrated in Figure 2, with a pattern σ of update sequences where $\sigma(e_4)$ and $\sigma(e_7)$ are arbitrary. Even if the update patterns for all remaining relations are insertion-only, we can still identify the following subquery $R_4(x_5, x_4) \bowtie R_3(x_4, x_1) \bowtie R_1(x_1, x_2, x_3) \bowtie R_5(x_2, x_3) \bowtie R_7(x_3, x_7)$ to simulate Q_{path}, after removing e_2, e_6, e_8, e_9 during the simplification step.

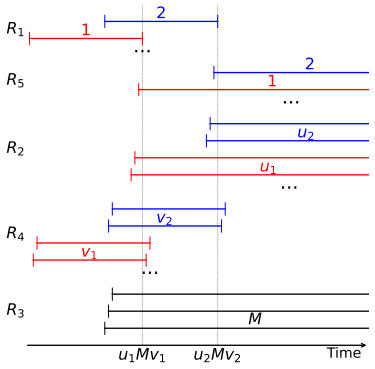


Figure 12. A reduction from an OuMv instance to a σ-sequence for Q_{path}, where $\sigma(\{e_1\})$ and $\sigma(\{e_4\})$ are not insertion-only.

Similar to Example 4.6, for any given instance of OuMv, we encode the matrix by R_1 and all pairs of vectors using two subqueries $\pi_{x_1} R_4(x_5, x_4) \bowtie R_3(x_4, x_1)$ and $\pi_{x_2} R_5(x_2, x_3) \bowtie R_7(x_3, x_7)$, respectively. We set the domains of x_1 and x_3 to $[n]$, and set the domain of any remaining attributes to $\{*\}$ for a special value $*$. For all the remaining relations, we insert all possible tuples across their domains at the beginning. Consequently, there is a one-to-one correspondence between the query result of Q_1 and that of Q_{path}. Hence, the hardness result for Q_1 follows Example 4.6.

4.3 Base Case 2: Line 4 of Algorithm 2

Lemma 4.8. *For a weak-q-hierarchical CQ Q and a pattern σ of update sequences, to which no simplification step can be applied if Q is weak-q-hierarchical but does not have a free-connex join tree compatible with σ, Q cannot be maintained in $O\left(\sqrt{|D|}^{1-\epsilon}\right)$ time over σ-sequences for arbitrary $\epsilon > 0$, assuming the OuMv and OMv conjectures.*

If Q is q-hierarchical, it has a height-1 free-connex join tree $\mathcal{T}$ in which each leaf node has its parent as a generalized relation. Consequently, $\mathcal{T}$ is compatible with an arbitrary update sequence. In what follows, we focus on weak-q-hierarchical but non-q-hierarchical CQs. We now revisit the three simplest example CQs whose update sequences follow the pattern falling into base case 2.

Example 4.9. For $Q_{\text{qcore}} = \pi_{x_1} R_1(x_1, x_2) \bowtie R_2(x_2)$ and a pattern σ of update sequences, such that $\sigma(\{e_2\})$ is not insertion-only and $\sigma(\{e_1, e_2\})$ is not FIFO, no index can be updated in $O(\sqrt{|D|}^{1-\epsilon})$ amortized time over σ-sequences, while supporting $O(\sqrt{|D|}^{1-\epsilon})$-delay enumeration for any $\epsilon > 0$, assuming the OMv conjecture. Q_{qcore} only has one free-connex join tree $\mathcal{T}$ where R_1 is the root with R_2 as the child node. As $\mathcal{T}$ is not compatible with σ, $\sigma(\{e_2\})$ could be FIFO or arbitrary while $\sigma(\{e_1, e_2\})$ is arbitrary. The hardness follows [7]. See Figure 13 for an illustration.

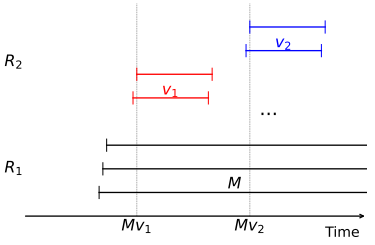


Figure 13. A reduction from an OMv instance to a σ-sequence for Q_{qcore}, where $\sigma(\{e_2\})$ is not insertion-only and $\sigma(\{e_1, e_2\})$ is not FIFO.

Example 4.10. For $Q_{\text{hier}} = \pi_{\varnothing} R_1(x_1, x_2) \bowtie R_2(x_2) \bowtie R_3(x_1)$ and a pattern σ of update sequences such that $\sigma(\{e_2\})$ and $\sigma(\{e_3\})$ are not insertion-only; and $\sigma(\{e_1, e_2\})$ and $\sigma(\{e_1, e_3\})$ are not FIFO, no index can be updated in $O(\sqrt{|D|}^{1-\epsilon})$ amortized time over σ-sequences, while supporting $O(\sqrt{|D|}^{2-\epsilon})$-delay enumeration for arbitrary $\epsilon > 0$, assuming the OuMv conjecture. Note that Q_{hier} only has three free-connex join trees: (1) $\mathcal{T}_1$ is rooted at R_1 with R_2, R_3 as its child node; (2) $\mathcal{T}_2$ is rooted at R_2 with R_1 as the child node of R_2, and R_3 as the child node of R_1; (3) $\mathcal{T}_3$ is rooted at R_3 with R_1 as the child node of R_3, and R_2 as the child node of R_1. As none of $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ is not compatible with σ, it must be that $\sigma(\{e_2\})$ and $\sigma(\{e_3\})$ are FIFO or arbitrary; and $\sigma(\{e_1, e_2\})$ and $\sigma(\{e_1, e_3\})$ are arbitrary. The hardness result then follows [7]. See Figure 14 for an illustration.

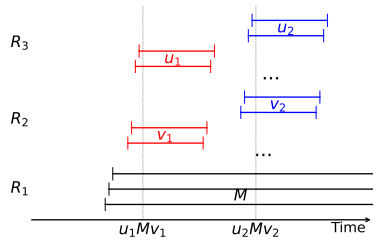


Figure 14. A reduction from an OuMv instance to a σ-sequence for Q_{hier}, where $\sigma(\{e_2\}), \sigma(\{e_3\})$ are not insertion-only, and $\sigma(\{e_1, e_2\}), \sigma(\{e_1, e_3\})$ are not FIFO.

Example 4.11. Consider $Q_{\text{swing}} = \pi_{\varnothing} R_1(x_1) \bowtie R_2(x_1, x_2) \bowtie R_3(x_2, x_3) \bowtie R_4(x_3)$ and a pattern σ of update sequences such that both $\sigma(\{e_1\}), \sigma(\{e_4\})$ are not insertion-only, and $\sigma(\{e_1, e_2\})$ is not FIFO; and (2) both $\sigma(\{e_1, e_2\}), \sigma(\{e_3, e_4\})$ are not FIFO. We first point out that no simplification step can be applied, i.e., no relation can be removed. In any free-connex join tree for Q_{swing}, we observe that either R_2 must be the

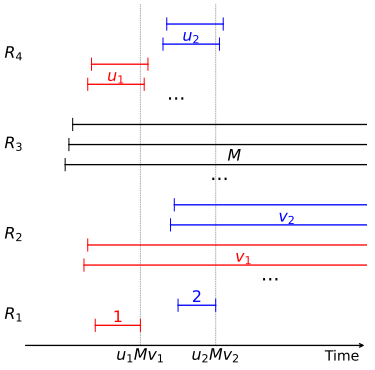


Figure 15. A reduction from an OuMv instance to a σ-sequence for Q_{swing}, where $\sigma(\{e_1\})$, $\sigma(\{e_4\})$ are not insertion-only and $\sigma(\{e_1, e_2\})$, $\sigma(\{e_3, e_4\})$ are not FIFO.

Algorithm 3: CompatibleTree(Q, σ)

1 $\mathcal{H} \leftarrow$ the query-update graph for Q and σ;
2 $\mathcal{E}_\circ \leftarrow$ the set of nodes with in-degree 0 in $\mathcal{H}$;
3 $\mathcal{T} \leftarrow$ a height-1 free-connex join tree for $\mathcal{E}_\circ$-induced CQ;
4 All nodes in $\mathcal{H}$ are marked as non-visited;
5 **while** *there exists some node in* $\mathcal{E}_\circ$ *non-visited* **do**
6 Visit an arbitrary non-visited node e in $\mathcal{E}_\circ$;
7 $Z \leftarrow \{e\}$;
8 **while** Z *changes from last iteration* **do**
9 **if** $\exists (e_1, e_2) \in \mathcal{H}$, $e_1 \in Z$ *and* e_2 *is non-visited* **then**
10 Add e_2 as a child of e_1 in $\mathcal{T}$;
11 Mark e_2 as visited;
12 $Z \leftarrow Z \cup \{e_2\}$;
13 **return** $\mathcal{T}$;

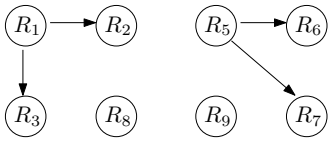


Fig. 16. Query-update graph of Q_4.

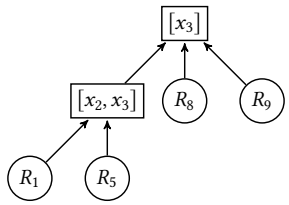


Fig. 17. The height-1 free-connex join tree for $\mathcal{E}_0$.

parent of R_1, or R_3 must be the parent of R_4. Thus, there exists no free-connex join tree compatible with σ and no index can be updated in $O(\sqrt{|D|}^{1-\epsilon})$ amortized time over σ-sequences, while supporting $O(\sqrt{|D|}^{2-\epsilon})$-delay enumeration for any $\epsilon > 0$, assuming the OuMv conjecture.

Given an instance of OuMv, we use R_3 to encode the matrix and use $R_5 = \pi_{x_2} R_1(x_1) \bowtie R_2(x_1, x_2)$ and R_4 to encode all pairs of vectors. We essentially maintain $Q_{\text{hier}} = \pi_\emptyset R_5(x_2) \bowtie R_3(x_2, x_3) \bowtie R_4(x_3)$ over σ'-sequences, where $\sigma'(\{e_5\})$ and $\sigma'(\{e_4\})$ are not insertion-only; and $\sigma'(\{e_5, e_3\})$ and $\sigma'(\{e_4, e_3\})$ are not FIFO since $\sigma(\{e_3\})$ is insertion-only. Hence, the hardness result follows Example 4.10. See Figure 15 for an illustration.

By gaining some intuition, we proceed to the general case. We begin by introducing the concept of a *query-update graph* $\mathcal{H}$ for a weak-q-hierarchical CQ Q and a pattern σ of update sequences. Let $\mathcal{E}_\clubsuit$ be a skeleton of Q. In a (directed) query-update graph $\mathcal{H}$, every relation $e \in \mathcal{E}$ is a node, and there is an edge from e' to e if the following conditions hold: (1) $e \in \mathcal{E} - \mathcal{E}_\clubsuit$, (2) e' is an anchor of e, and (3) either $\sigma(\{e\})$ is insertion-only or $\sigma(\{e, e'\})$ is FIFO. Let $\mathcal{E}_\circ$ denote the set of relations with in-degree as 0 in $\mathcal{H}$. Suppose we are given a weak-q-hierarchical CQ Q and a pattern σ of update sequences, to which no simplification step can be applied. Moreover, Q is weak-q-hierarchical but does not have a free-connex join tree compatible with σ. Below, we prove Lemma 4.8 through two steps: (1) the $\mathcal{E}_\circ$-induced CQ is not q-hierarchical (as shown in Lemma 4.12); and (2) Q cannot be maintained in $O\left(\sqrt{|D|}^{1-\epsilon}\right)$ time over σ-sequences for any $\epsilon > 0$ (as shown in Lemma 4.14).

Lemma 4.12. *$\mathcal{E}_\circ$-induced CQ is not q-hierarchical.*

We prove Lemma 4.12 by contradiction. More specifically, if $\mathcal{E}_\circ$-induced CQ is q-hierarchical, then Q has a free-connex join tree compatible with σ, leading to contradiction. We present a procedure in Algorithm 3 to construct a free-connex compatible join tree with σ for Q. We begin by building a height-1 free-connex join tree $\mathcal{T}$ for $\mathcal{E}_\circ$ (lines 2-3), which is always feasible from Lemma 2.2. Next, we incrementally add relations from $\mathcal{E} - \mathcal{E}_\circ$ to $\mathcal{T}$. Initially, all relations are non-visited (line 4). We first visit an arbitrary relation $e \in \mathcal{E}_\circ$ (lines 6-7) and then explore all non-visited nodes reachable from e in $\mathcal{H}$ (lines 8-12). If a node e' is visited for the first time from e'', we add e' as a child of e''. Once all nodes reachable from e have been visited, we proceed to the next unvisited relation in $\mathcal{E}_\circ$, repeating this process until all nodes in $\mathcal{E}_\circ$ have been traversed. For each $e \in \mathcal{E}_\circ$, the subtree of $\mathcal{T}$

rooted at e satisfies the following condition: for any node $e' \in \mathcal{E} - \{e\}$, its parent node e'', must be an anchor of e', and either $\sigma(\{e'\})$ is insertion-only, or $\sigma(\{e', e''\})$ is FIFO. Consequently, $\mathcal{T}$ is a free-connex join tree compatible with σ.

Example 4.13. Figure 16 illustrates the query-update graph of Q_4 from Example 4.2. $\mathcal{E}_\circ = \{R_1, R_5, R_8, R_9\}$ induces a q-hierarchical CQ $R_1(x_1, x_2, x_3) \bowtie R_5(x_2, x_3) \bowtie R_8(x_3, x_8) \bowtie R_9(x_3, x_9)$. We first construct a height-1 free-connex join tree for $\mathcal{E}_\circ$, shown in Figure 17. As we traverse $\mathcal{H}$, we incrementally add relations: R_2 and R_3 are attached as child nodes of R_1, while R_6 and R_7 are added as two child nodes of R_5. The final free-connex join tree for Q is depicted in Figure 9.

Lemma 4.14. *If $\mathcal{E}_\circ$-induced CQ is not q-hierarchical, Q cannot be maintained in $O\left(\sqrt{|D|}^{1-\epsilon}\right)$ time over σ-sequences for any $\epsilon > 0$.*

Lemma 4.15 ([7]). *Any non-q-hierarchical CQ $Q = (\mathcal{V}, \mathcal{E}, \mathbf{y})$ must contain one of the following structures:*

- **(q-core)** *relations $e_1, e_2 \in \mathcal{E}$ and attributes $x_1 \in \mathbf{y} \cap (e_1 - e_2)$, $x_2 \in \bar{\mathbf{y}} \cap e_1 \cap e_2$.*
- **(hierarchical core)** *relations $e_1, e_2, e_3 \in \mathcal{E}$ and attributes $x_1 \in (e_2 - e_3) \cap e_1$, $x_2 \in (e_3 - e_2) \cap e_1$.*

Proof of Lemma 4.14. As the $\mathcal{E}_\circ$-induced CQ is not q-hierarchical, it contains either a q-core or a hierarchical core by Lemma 4.15. We first assume a q-core with relations e_1, e_2 and attributes $x_1 \in \mathbf{y} \cap (e_1 - e_2), x_2 \in \bar{\mathbf{y}} \cap e_1 \cap e_2$. As $Q_\clubsuit$ is q-hierarchical, the case with $e_1, e_2 \in \mathcal{E}_\clubsuit$ cannot happen. We further distinguish the following three cases:

- **Case 1.1:** $e_1 \in \mathcal{E}_\clubsuit$ and $e_2 \in \mathcal{E}_\circ - \mathcal{E}_\clubsuit$. There must exist a relation $e_3 \in \mathcal{E}_\clubsuit$ that is an anchor of e_2 and contains both attributes x_1, x_2. If no such relation exists, for any anchor $e_3 \in \mathcal{E}_\clubsuit$ of e_2, we have $x_2 \in e_1 - \mathcal{V}_\bullet \subseteq e_3$ and therefore $x_1 \in e_1 - e_3$, which together forms a q-core with relations e_1, e_3 and attributes x_1, x_2 in $Q_\clubsuit$. This contradicts the fact that $Q_\clubsuit$ is q-hierarchical. After finding such a relation e_3 with $x_1, x_2 \in e_3$, we identify another q-core with relations e_2, e_3 and attributes $x_1 \in \mathbf{y} \cap (e_3 - e_2), x_2 \in \bar{\mathbf{y}} \cap e_2 \cap e_3$. As there is no edge between e_2, e_3 in $\mathcal{H}$, $\sigma(\{e_2\})$ is not insertion-only and $\sigma(\{e_2, e_3\})$ is not FIFO. The hardness follows Example 4.9.
- **Case 1.2:** $e_1, e_2 \in \mathcal{E}_\circ - \mathcal{E}_\clubsuit$. For any anchor $e_3 \in \mathcal{E}_\clubsuit$ of e_1, as e_1 is reducible by e_3, $x_2 \in e_1 - \mathcal{V}_\circ \subseteq e_3$. Moreover, $x_2 \in \bar{\mathbf{y}}$, hence $x_1 \in \mathbf{y} \cap e_1 \subseteq e_3$. We can apply the same argument as **Case 11.** with relations e_2, e_3 and attributes x_1, x_2.
- **Case 1.3:** $e_1 \in \mathcal{E}_\circ - \mathcal{E}_\clubsuit$ and $e_2 \in \mathcal{E}_\clubsuit$. By applying the same argument as **Case 1.1**, we can identify an anchor $e_3 \in \mathcal{E}_\clubsuit$ of e_1 with $x_1, x_2 \in e_3$. Now, we have identified a q-core in $Q_\clubsuit$ with relation e_3, e_2 and attributes x_1, x_2, contradicting the fact that $Q_\clubsuit$ is q-hierarchical.

Alternatively, we assume a hierarchical core in the $\mathcal{E}_\circ$-induced CQ with relations e_1, e_2, e_3 and attributes $x_1 \in (e_2 - e_3) \cap e_1, x_2 \in (e_3 - e_2) \cap e_1$. If $e_1 \notin \mathcal{E}_\clubsuit$, we find an arbitrary anchor $e_1' \in \mathcal{E}_\clubsuit$ of e_1. As e_1 is reducible by e_1', we have $e_1 - \mathcal{V}_\bullet \subseteq e_1'$ and furthermore $x_1, x_2 \in e_1'$. We replace e_1 with e_1'. As $Q_\clubsuit$ is q-hierarchical, the case with $e_2, e_3 \in \mathcal{E}_\clubsuit$ cannot happen. We further distinguish two cases:

- **Case 2.1:** $e_2 \in \mathcal{E}_\circ - \mathcal{E}_\clubsuit$ and $e_3 \in \mathcal{E}_\clubsuit$ (the case with $e_3 \in \mathcal{E}_\circ - \mathcal{E}_\clubsuit$ and $e_2 \in \mathcal{E}_\clubsuit$ is symmetric). As $e_1, e_3 \in \mathcal{E}_\clubsuit$, we must have $e_3 - e_1 \neq \emptyset$. We can always identify an attribute $x_3 \in e_3 - e_1$ satisfying the following two properties:
  - $\mathcal{E}_{x_3} - \mathcal{E}_\clubsuit \neq \emptyset$: By contradiction, assume each attribute $x_3 \in e_3 - e_1$ satisfying $\mathcal{E}_{x_3} - \mathcal{E}_\clubsuit = \emptyset$, which must be a joint attribute. For each such attribute $x_3 \in e_3 - e_1 - \mathcal{V}_\bullet$, as $\mathcal{E}_{x_3} \subseteq \mathcal{E}_\clubsuit$, there exists a relation $e'_{x_3} \in \mathcal{E}_\clubsuit$ such that $x_3 \subseteq e'_{x_3}$. Now, if there exists a pair of $x_3, x_4 \in e_3 - e_1 - \mathcal{V}_\bullet$ such that $e'_{x_3} \neq e'_{x_4}$, we have $e_3 \in \mathcal{E}_{x_3} \cap \mathcal{E}_{x_4} \cap \mathcal{E}_\clubsuit \neq \emptyset$, $e'_{x_3} \in \mathcal{E}_{x_3} \cap \mathcal{E}_\clubsuit - \mathcal{E}_{x_4}$ and $e'_{x_4} \in \mathcal{E}_{x_4} \cap \mathcal{E}_\clubsuit - \mathcal{E}_{x_3}$, contradicting the fact that $Q_\clubsuit$ is q-hierarchical. Hence, there is a relation e' such that $e_3 - e_1 - \mathcal{V}_\bullet \subseteq e'$. For each attribute $x_5 \in e_3 \cap e_1$, $x_5 \in e'$ must hold; otherwise, we have $e_3 \in \mathcal{E}_{x_5} \cap \mathcal{E}_{x_3} \cap \mathcal{E}_\clubsuit \neq \emptyset$, $e' \in \mathcal{E}_{x_3} \cap \mathcal{E}_\clubsuit - \mathcal{E}_{x_5}$ and $e_1 \in \mathcal{E}_{x_5} \cap \mathcal{E}_\clubsuit - \mathcal{E}_{x_3}$, again contradicting

the fact that $Q_\clubsuit$ is q-hierarchical. Hence, $e_3 - \mathcal{V}_\bullet \subseteq e'$, i.e., e_3 is reducible, contradicting that fact that $e_3 \in \mathcal{E}_\clubsuit$.
- $\mathcal{E}_{x_1} \cap \mathcal{E}_{x_3} = \emptyset$: Suppose there exists any $e \in \mathcal{E}_\clubsuit \cap \mathcal{E}_{x_1} \cap \mathcal{E}_{x_3}$, then we identify a hierarchical core with relations e_1, e_3, e and attributes $x_1 \in (e_1 - e_3) \cap e, x_2 \in (e_3 - e_1) \cap e$, contradicting the fact that $Q_\clubsuit$ is q-hierarchical. Otherwise, suppose there exists some $e \in \mathcal{E}_\circ - \mathcal{E}_\clubsuit \cap \mathcal{E}_{x_1} \cap \mathcal{E}_{x_3}$. For any anchor $e' \in \mathcal{E}_\clubsuit$ of e, $x_1, x_3 \in e - \mathcal{V}_\bullet \subseteq e'$. However, $\mathcal{E}_\clubsuit \cap \mathcal{E}_{x_1} \cap \mathcal{E}_{x_3} = \emptyset$ by our assumption. Hence, e does not have an anchor, violating the condition that e is reducible.

Let $e_4 \in \mathcal{E}_\circ - \mathcal{E}_\clubsuit \cap \mathcal{E}_{x_3}$ be any relation. We can reduce Q_{swing} to Q with attributes x_1, x_2, x_3 and relations e_1, e_2, e_3, e_4. As there is no edges (e_1, e_2) and (e_3, e_4) in $\mathcal{H}$, either $\sigma(\{e_1\})$ or $\sigma(\{e_4\})$ is not insertion-only, and $\sigma(\{e_1, e_2\})$ and $\sigma(\{e_3, e_4\})$ are not FIFO. The hardness follows Example 4.11.

- **Case 2.2:** $e_2, e_3 \in \mathcal{E}_\circ - \mathcal{E}_\clubsuit$. Let $e_4 \in \mathcal{E}_\clubsuit$ be an arbitrary anchor of e_2 and $e_5 \in \mathcal{E}_\clubsuit$ be an arbitrary anchor of e_3. Hence, $x_1 \in e_2 - \mathcal{V}_\bullet \subseteq e_4$ and $x_2 \in e_3 - \mathcal{V}_\bullet \subseteq e_5$. Additionally, either $x_2 \in e_4$ or $x_1 \in e_5$; otherwise, we identify a hierarchical core, contradicting the fact that $Q_\clubsuit$ is q-hierarchical. Moreover, if $x_2 \in e_4$ but $x_1 \notin e_5$ (the case with $x_1 \in e_5$ but $x_2 \notin e_4$ is symmetric), we can reduce e_4, e_2, e_5 to **Case 2.1**. Below, we assume $x_2 \in e_4, x_1 \in e_5$, and further distinguish the following two cases:
  - **Case 2.2.1:** $e_4 = e_5$. In this case, we identify a hierarchical core e_4, e_2, e_3. Here, $\sigma(\{e_2\})$ and $\sigma(\{e_3\})$ are not insertion-only, and $\sigma(\{e_4, e_2\})$ and $\sigma(\{e_4, e_3\})$ are not FIFO, as there is no edge between (e_4, e_2) and (e_4, e_3) in $\mathcal{H}$. The hardness follows Example 4.10.
  - **Case 2.2.2:** e_2, e_3 don't share a common anchor. We can find $x_3 \in e_2 - \mathcal{V}_\bullet - e_5$ and $x_4 \in e_3 - \mathcal{V}_\bullet - e_4$. We can reduce Q_{swing} to Q with attributes x_3, x_1, x_4 and relations e_2, e_4, e_5, e_3. As there is no edge (e_4, e_2) and (e_5, e_3) in $\mathcal{H}$, either $\sigma(\{e_2\})$ or $\sigma(\{e_3\})$ is not insertion-only, and $\sigma(\{e_2, e_4\})$ and $\sigma(\{e_5, e_3\})$ are not FIFO. The hardness follows Example 4.11. □

4.4 Base Case 3: Line 7 of Algorithm 2

Lemma 4.16. *For a CQ Q and a pattern σ of update sequences, to which no simplification step can be applied if Q is weak-q-hierarchical and has a free-connex join tree $\mathcal{T}$ compatible with σ, Q can be maintained in $O(1)$ amortized time over σ-sequences.*

In Appendix B, we show that for a CQ Q and a pattern σ of update sequences, if Q is weak-q-hierarchical and has a free-connex join tree $\mathcal{T}$ compatible with σ, any σ-sequence S has enclosureness $O(1)$ under $\mathcal{T}$. Together with Lemma 1.4, we complete the proof of Lemma 4.16.

5 CONCLUSION

In this paper, we study the lower bounds of query maintenance over different classes of update sequences. There are a few exciting open questions in this direction:

- *Single update v.s. Batch update.* So far, we have focused on the maintenance complexity for single-tuple updates. In practice, it is more likely that updates arrive in batch, and query answers are only required to be ready to be enumerated after each batch. A strawman approach is to drop the single-update assumption for updates and update sequences, then apply the current upper bounds. It would be very interesting to study the tradeoff between maintenance complexity, delay, and batch size beyond such an approach.
- *Preprocessing.* So far, we simulate the preprocessing steps with an insertion-only update sequence prior to time 0. The absence of strict update ordering at the start allows for linear-time preprocessing for free-connex queries (also shown by [17]). For non-free-connex or cyclic queries, strategically assigning preprocessing order could improve performance, and exploring the corresponding upper and lower bounds would be an interesting direction for future work.

- *Better update-independent quantity beyond enclosureness.* The current notion of enclosureness is sometimes too loose to capture the hardness of update sequences. In Appendix B, we have to slightly modify the data structure in [25] to derive a tighter analysis. It is open whether one can find a better quantity than enclosureness to characterize the hardness of query maintenance.
- *The effect of self-joins.* It has been observed [10] in static query enumeration that self-joins will change the existing complexity results established for a long time. As mentioned, there is a natural connection between static query enumeration and query maintenance under insertion-only updates. In general, it remains to investigate how self-joins would change the problem hardness of query maintenance.

ACKNOWLEDGEMENT

This work was supported by NSERC – Discovery Grant and Hong Kong RGC Grants (Project No. 12200524, C2004-21GF, and C2003-23Y).

A MISSING PROOFS IN SECTION 3

Lemma A.1. *For a free-connex but non-weak-q-hierarchical CQ $Q = (\mathcal{V}, \mathcal{E}, \mathbf{y})$ with a skeleton $\mathcal{E}_\clubsuit$, if the $\mathcal{E}_\clubsuit$-induced CQ contains a q-core, then Q must contain a hook core.*

Proof of Lemma A.1. By definition, there must exist a pair of distinct attributes $x_1 \in \mathbf{y}, x_2 \in \bar{\mathbf{y}}$ and a pair of distinct relations $e_1, e_2 \subseteq \mathcal{E}_\clubsuit$ such that $x_1 \in e_1 - e_2$ and $x_2 \in e_1 \cap e_2$. Consider an arbitrary free-connex tree $\mathcal{T}$ for Q, whose internal (original) nodes are $\mathcal{E}_\clubsuit$. Let $\mathcal{E}_{\mathsf{con}}$ be the connex subtree of $\mathcal{T}$. We first show that e_2 cannot be the ancestor of e_1. If this is the case, all nodes containing x_1 must be a subtree of x_2. Let e be the child node of e_2 lying on the path from e_2 to e_1. Hence, $e_2, e \in \mathcal{E}_{\mathsf{con}}$; otherwise, no node in $\mathcal{E}_{\mathsf{con}}$ contains x_1. However, $x_2 \in e_2 \cap e = \mathsf{key}(e) \notin \mathbf{y}$, coming to a contradiction of the definition of $\mathcal{E}_{\mathsf{con}}$. Let e_5 be the parent node of e_2. We claim that $x_2 \in e_5$; otherwise, e_1 must be a descendent of e_2, contradicting our observation above. Also, $x_1 \in e_5$; otherwise, the connect property of x_1 does not hold. As $e_2 \cap e_5 \cap \bar{\mathbf{y}} \neq \emptyset$, then $e_2 \notin \mathcal{E}_{\mathsf{con}}$. Then, $(e_2 - e_5) \cap \mathbf{y} = \emptyset$; otherwise, the connect property of any output attribute $x \in (e_2 - e_5) \cap \mathbf{y}$ does not hold. As $e_2 \in \mathcal{E}_\clubsuit$ and $(e_2 - e_5) \cap \mathbf{y} = \emptyset$, we must have $(e_2 - \mathcal{V}_\bullet - e_5) \cap \bar{\mathbf{y}} \neq \emptyset$; otherwise, e_2 can be removed from $\mathcal{E}_\clubsuit$, due to the existence of e_5. Let $x_3 \in (e_2 - \mathcal{V}_\bullet - e_5) \cap \bar{\mathbf{y}}$ be an arbitrary non-output attribute. As $x_3 \notin \mathcal{V}_\bullet$, there must exist a child node of e_2 say e_3 in $\mathcal{T}$, such that $x_3 \in e_2 \cap e_3$. As $x_1 \notin e_2, x_1 \notin e_3$. As $x_3 \notin e_5, x_3 \notin e_1$. Also, there exists no relation containing both x_1, x_3. Hence, we have identified a hook core with x_1, x_2, x_3 and e_1, e_2, e_3 as desired. □

Lemma A.2. *For a free-connex CQ $Q = (\mathcal{V}, \mathcal{E}, \mathbf{y})$, if it does not contain a q-core, $\mathcal{V}_\bullet \subseteq \bar{\mathbf{y}}$ or $\bar{\mathbf{y}} \subseteq \mathcal{V}_\bullet$.*

Proof of Lemma A.2. It suffices to show that if $y \cap \mathcal{V}_\bullet = \emptyset$, then $\mathcal{V}_\bullet \subseteq \bar{\mathbf{y}}$. Suppose $x \in \mathbf{y} \cap \mathcal{V}_\bullet$ with $x \in e$ for the unique $e \in \mathcal{E}$. We start with relation e, and add it to X. We keep adding any relation e' that shares some common attribute with any $e \in X$ to X, until we find some e' with $e'' \cap e' \cap \bar{\mathbf{y}} \neq \emptyset$ for any $e'' \in X$, or all relations in $\mathcal{E}$ have been added to X. Suppose all relations have been added without finding such a relation e'. Then, $\mathcal{V} - \mathcal{V}_\bullet \subseteq \mathbf{y}$, i.e., $\bar{\mathbf{y}} \subseteq \mathcal{V}_\bullet$. Suppose such a relation e' is identified. Let $e'' \in X$ be such a relation that $e' \cap e'' \cap \bar{\mathbf{y}} \neq \emptyset$. Let $x_1 \in e'' \cap \mathbf{y} - e'$ and $x_2 \in e' \cap e'' \cap \bar{\mathbf{y}}$. Then, we have found a q-core with x_1, x_2 and e'', e', coming to a contradiction. □

Lemma A.3. *For a free-connex but non-weak-q-hierarchical CQ Q with a skeleton $\mathcal{E}_\clubsuit$, if the $\mathcal{E}_\clubsuit$-induced CQ does not contain a q-core but a hierarchical core, Q must contain a path core.*

Proof of Lemma A.3. As Q contains a hierarchical core, there must exist a pair of attributes x_2, x_3 and three relations $e_2, e_3, e_4 \in \mathcal{E}_\clubsuit$ such that $x_2 \in e_2 \cap e_3 - e_4$ and $x_3 \in e_3 \cap e_4 - e_2$. Consider an arbitrary free-connex $\mathcal{T}$ for Q, whose internal (original) nodes exactly correspond to $\mathcal{E}_\clubsuit$. Let $\mathcal{E}_{\mathsf{con}}$

be the connex subtree of $\mathcal{T}$. We start with two critical observations: If e_2 is not the root of $\mathcal{T}$, let e_2' be the parent of e_2. We argue that $e_2 - e_2' - \mathcal{V}_\bullet \neq \emptyset$ by distinguishing the following two cases:

- if $e_2 \cap \mathcal{V}_\bullet \subseteq \bar{\mathbf{y}}$, then $e_2 - e_2' - \mathcal{V}_\bullet \neq \emptyset$; otherwise, e_2 can be removed from $\mathcal{E}_\clubsuit$ due to e_2';
- if $e_2 \cap \mathcal{V}_\bullet \cap \mathbf{y} \neq \emptyset$, then $e_2', e_2 \in \mathcal{E}_{\mathsf{con}}$; otherwise, no relation in $\mathcal{E}_{\mathsf{con}}$ contains attributes in $e_2 \cap \mathcal{V}_\bullet \cap \mathbf{y}$, contradicting the property of $\mathcal{E}_{\mathsf{con}}$. From the facts that $e_2', e_2 \in \mathcal{E}_{\mathsf{con}}$, we observe $e_2' \cap e_2 \subseteq \mathbf{y}$ and therefore $e_2 - e_2' - \mathcal{V}_\bullet \neq \emptyset$; otherwise, $e_2 - \mathcal{V}_\bullet \subseteq e_2' \cap \mathbf{y}$ and e_2 can be removed from $\mathcal{E}_\clubsuit$ due to the existence of e_2'.

Similarly, if e_4 is not the root of $\mathcal{T}$, let e_4' be the parent of e_4. We argue that $e_4 - e_4' - \mathcal{V}_\bullet \neq \emptyset$. We are ready to prove Lemma A.3 by distinguishing two cases: **(Case 1)** e_2 is not the ancestor of e_4, nor e_4 is not the ancestor of e_2. **(Case 2)** e_2 is the ancestor of e_4 or e_4 is the ancestor of e_2.

In **(Case 1)**, neither e_2 nor e_4 is the root of $\mathcal{T}$. In that case, e_3 must not be a decedent of e_2 or e_4; otherwise, the connect property of x_2 or x_3 is violated. Let e_2', e_4' be the parent of e_2, e_4 respectively. Implied by the connect property of x_2 and x_3 again, $x_2 \in e_2'$ and $x_3 \in e_4'$. After establishing $e_2 - e_2' - \mathcal{V}_\bullet \neq \emptyset$, we identify $x_1 \in e_2 - e_2' - \mathcal{V}_\bullet$ as such an arbitrary attribute. Noted that $x_1 \neq x_2$ as $x_2 \in e_2'$. Some child node e_1 of e_2 must contain x_1. The e_5 and x_4 can be identified similarly. As $x_3 \notin e_2$, no relation contains both x_1, x_3. As $x_4 \notin e_3$, no relation contains both x_1, x_4, or both x_2, x_4. Hence, we have identified a path core with x_1, x_2, x_3, x_4 and e_1, e_2, e_3, e_4, e_5.

In **(Case 2)**, assume e_4 is an ancestor of e_2 wlog. Implied by the property of free-connex tree $\mathcal{T}$, e_4 is an ancestor of e_3, and e_3 is an ancestor of e_2. Let e_2' be the parent of e_2. Again, we have $e_2 - e_2' - \mathcal{V}_\bullet \neq \emptyset$, and identify x_1, e_1 as **(Case 1)**. We next focus on how to identify e_5 and x_4. Let e_4' be the child node of e_4 lying on the path from e_4 to e_2. Note that e_4' may be the same as e_3.

We next argue that $e_4 - e_4' - \mathcal{V}_\bullet \neq \emptyset$. If $e_4 \cap \mathcal{V}_\bullet \subseteq \bar{\mathbf{y}}$, we must have $e_4 - e_4' - \mathcal{V}_\bullet \neq \emptyset$; otherwise, e_4 can be removed from $\mathcal{E}_\clubsuit$ due to the existence of e_4'. Then, it suffices to consider the case with $e_4 \cap \mathcal{V}_\bullet \cap \mathbf{y} \neq \emptyset$. Implied by Lemma A.2, $x_2 \in \mathbf{y}$. As $x_2 \notin e_4$, we must have $e_4, e_4' \in \mathcal{E}_{\mathsf{con}}$; otherwise, no relation in $\mathcal{E}_{\mathsf{con}}$ contains attribute x_2, coming to a contradiction. Implied by the definition of $\mathcal{E}_{\mathsf{con}}$, $e_4 \cap e_4' \subseteq \mathbf{y}$. Hence, $e_4 - e_4' - \mathcal{V}_\bullet \neq \emptyset$; otherwise, $e_4 - \mathcal{V}_\bullet \subseteq e_4' \cap \mathbf{y}$, and e_4 can be removed from $\mathcal{E}_\clubsuit$, due to the existence of e_4', coming a contradiction.

After establishing $e_4 - e_4' - \mathcal{V}_\bullet \neq \emptyset$, we identify $x_4 \in e_4 - e_4' - \mathcal{V}_\bullet$ as such an arbitrary attribute. There must exist some child node e_5 of e_4, or the parent node of e_4 that contains x_4. Noted that $x_2 \notin e_5$, since $x_2 \in e_2 \cap e_3 - e_4$, together with the connect property of x_2. As $x_3 \notin e_2$ and x_1 only appear in relations that are descendants of e_2, no relation contains both x_1, x_3. As $x_4 \notin e_4'$, we have $x_4 \notin e_3$, and no relation contains both x_1, x_4, or both x_2, x_4. Hence, we have identified a path core with x_1, x_2, x_3, x_4 and e_1, e_2, e_3, e_4, e_5. □

Proof of Lemma 3.11. Let $\mathcal{E}_\clubsuit$ be the skeleton of $\mathcal{E}$. Consider a free-connex tree $\mathcal{T}$ for Q, where $\mathcal{E}_\clubsuit$ is exactly the set of internal nodes of $\mathcal{T}$. As Q is free-connex but not weak-q-hierarchical, the $\mathcal{E}_\clubsuit$-induced CQ $Q_\clubsuit$ is not q-hierarchical. If $Q_\clubsuit$ contains a q-core, Q must contain a hook core, implied by Lemma A.1. If $Q_\clubsuit$ does not contain a q-core but a hierarchical core, Q must contain a path core, implied by Lemma A.3. □

Proof of Theorem 3.12. For any FIFO sequence S, we denote $\tau[S, \theta]$ as the first deletion of any tuple inserted after timestamp θ. More specially, there exists some tuple t such that $(t, s_1, +, *) \in S$ and $(t, \tau[S, \theta], -, *) \in S$ with $\theta \leqslant s_1$, but there exists no tuple t' such that $(t', s_2, +, *) \in S$ with $\theta \leqslant s_2 < \tau[S, \theta]$. If no tuple is inserted after timestamp θ, we set $\tau[S, \theta] = +\infty$. Implied by Lemma 3.11, a free-connex but non-weak-q-hierarchical query must contain **Case (1)** a path core or **Case (2)** a hook core.

Case (1.1): $x_2 \notin e_1$ **and** $x_3 \notin e_5$**.** Given an arbitrary FIFO sequence S' for Q_{path}, we construct a FIFO sequence S for Q. For every relation $e \in \mathcal{E}$, we assign i as the unique integer in advance to ensure that at each timestamp, there can be at most one update, and

- if $e \cap \{x_1, x_2, x_3, x_4\} = \emptyset$, we add a tuple $(*)$ at timestamp $t_0 = -\infty$ and delete it at timestamp $t_1 = \tau[S', -\infty] + i\epsilon$. We reinsert $(*)$ at timestamp t_1 and delete it at timestamp $t_2 = \tau[S', t_1] + i\epsilon$. Again, we reinsert $(*)$ at timestamp t_2 and delete it at timestamp $t_3 = \tau[S', t_2] + i\epsilon$. We repeat this procedure until $\tau[S', t_k] = +\infty$. There are at most $|D|$ tuples in R_e.
- If $e \cap \{x_1, x_2, x_3, x_4\} \neq \emptyset$, we apply the same updates as S' does for $R_{e'}$, where $e' = e \cap \{x_1, x_2, x_3, x_4\}$. Suppose $e' = \{x_1\}$. When a tuple t is inserted into R_1 at timestamp t^+ and deleted at timestamp t^- in S', we insert a tuple t' such that $\pi_{x_1} t' = t$ and $\pi_x t' = \{*\}$ for any attribute $x \in e - \{x_1\}$, into R_e at timestamp $t^+ + i\epsilon$ and delete at timestamp $t^- + i\epsilon$ in S.

It remains to show that S is a FIFO sequence. For tuples in relations R_e with $e \cap \{x_1, x_2, x_3, x_4\} = \emptyset$, adding these updates won't violate the FIFO property, since during the lifespan $[t_i, t_{i+1}]$, there exists no tuple t' such that $(t', s_1, +, *) \in S'$ and $(t', s_2, -, *) \in S'$ with $t_i < s_i < s_2 < t_{i+1}$; otherwise $\tau[S', t_i] = s_2$, coming to a contradiction. Whenever an enumeration query is issued to Q_{path}, we ask an enumeration query to Q. It can be easily checked that there is a one-to-one correspondence between Q_{path} and Q at any timestamp.

Case (1.2): $x_2 \in e_1$ **or** $x_3 \in e_5$**.** If $x_2 \in e_1$, we select an arbitrary relation $e \in \mathcal{E}\{e_2\}$ with $e \cap \{x_1, x_2, x_3, x_4\} = \{x_1, x_2\}$ to simulate e_1, by applying all updates that S' does for R_1 to R_e; similar modification will be applied if $x_3 \in e_5$.

Case (2.1): $x_2 \notin e_3$**.** Given an arbitrary FIFO sequence S' for Q_{hook}, we construct a FIFO sequence S for Q. For every relation $e \in \mathcal{E}$,

- If $e \cap \{x_1, x_2, x_3\} = \emptyset$, we add a tuple $(*)$ at timestamp $t_0 = -\infty$ and delete it at timestamp $t_1 = \tau[S', -\infty] + i\epsilon$. We reinsert $(*)$ at timestamp t_1 and delete it at timestamp $t_2 = \tau[S', t_1] + i\epsilon$. Again, we reinsert $(*)$ at timestamp t_2 and delete it at timestamp $t_3 = \tau[S', t_2] + i\epsilon$. We repeat this procedure until $\tau[S', t_k] = +\infty$. There are at most $|D|$ tuples in R_e.
- If $e \cap \{x_1, x_2, x_3\} \neq \emptyset$, we apply the same updates as S' does for $R_{e'}$, where $e' = e \cap \{x_1, x_2, x_3\}$. Suppose $e' = \{x_1, x_2\}$. When a tuple t is inserted into R_1 at timestamp s in S', we insert a tuple t' such that $\pi_{x_1, x_2} t' = t$ and $\pi_x t' = \{*\}$ for any attribute $x \in e - \{x_1, x_2\}$, into R_e at timestamp $s + i\epsilon$ in S and delete accordingly.

It can be proved similarly that S is a FIFO sequence. Whenever an enumeration query is issued to Q_{hook}, we ask an enumeration query to Q. It can be easily checked that there is a one-to-one correspondence between Q_{hook} and Q at any timestamp.

Case (2.2): $x_2 \in e_3$**.** If $x_2 \in e_3$, similar to case (1.2), we select an arbitrary relation R_e with $e \cap \{x_1, x_2, x_3\} = \{x_2, x_3\}$, and apply all updates that S' does for R_3 to R_e. □

B MISSING DETAILS OF SECTION 4

The other three cases for Example 4.6 are illustrated in Figures 18 - 20.

Lemma B.1. *Algorithm 2 always returns the same answer no matter which ordering is applied to remove reducible relations.*

Proof of Lemma B.1. The *reducible* relationship can be modeled as a directed graph, where each vertex represents a relation e, and there exists a direct edge from e_1 to e_2 if e_1 is reducible with e_2 as the anchor, and both $\sigma(e_1), \sigma(e_2)$ are insertion-only. If the graph is acyclic, by running Algorithm 2, the residual query (after removing all possible reducible relations) is unique, with a determined final output. On the other hand, if there exist two relations e_1, e_2 that can be reducible by each other as an anchor, we must have $e_1 - \mathcal{V}_\bullet = e_2 - \mathcal{V}_\bullet$, and both $\sigma(e_1)$ and $\sigma(e_2)$ are insertion-only.

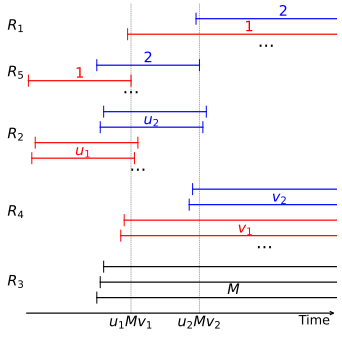


Fig. 18. A reduction from an OuMv instance to a σ-sequence for Q_{path}, where $\sigma(\{e_2\})$ and $\sigma(\{e_5\})$ are not insertion-only.

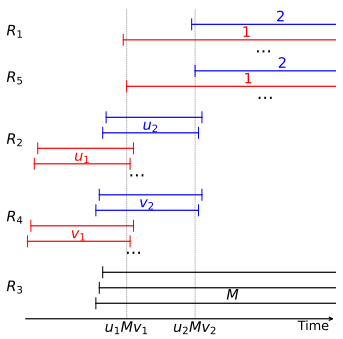


Fig. 19. A reduction from an OuMv instance to a σ-sequence for Q_{path}, where $\sigma(\{e_2\})$ and $\sigma(\{e_4\})$ are not insertion-only.

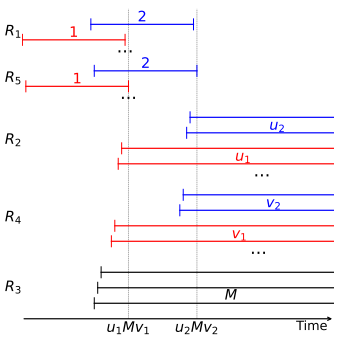


Fig. 20. A reduction from an OuMv instance to a σ-sequence for Q_{path}, where $\sigma(\{e_1\})$ and $\sigma(\{e_5\})$ are not insertion-only.

We can further distinguish two more cases. If an anchor e_3 exists for both e_1 and e_2, such that $\sigma(e_3)$ is also insertion-only. In this case, e_1 and e_2 will be removed by e_3, no matter in which order, without making any difference to the final output. Otherwise, either e_1 or e_2 remains in the residual query. In this case, the remaining relation will appear as a leaf node of any join tree. As both $\sigma(e_1)$ and $\sigma(e_2)$ are insertion-only and $e_1 - \mathcal{V}_\bullet = e_2 - \mathcal{V}_\bullet$, this remaining relation won't change the if-condition of Algorithm 2. Hence, Algorithm 2 returns the same answer. □

Proof of Lemma 4.4. Let $\mathcal{E}_\clubsuit$ be the skeleton of $\mathcal{E}$. Let $\mathcal{T}_\clubsuit$ be the free-connex join tree of $\mathcal{E}_\clubsuit$-induced CQ. We construct a free-connex join tree $\mathcal{T}$ for Q as follows. For every relation $e \in \mathcal{E} - \mathcal{E}_\clubsuit$, some relation $e' \in \mathcal{E}_\clubsuit$ must exist such that e' is an anchor of e. We just add e as a child of e'. From Lemma 3.11, as Q is not weak-q-hierarchical, we can always find a **path core** or **hook core**. In addition, we aim to identify either a **path core** where e_1 and e_5 are leaf or root nodes in the join tree $\mathcal{T}$, or a **hook core** where e_3 is a leaf node in $\mathcal{T}$. Since no simplification steps can be applied, for such a **path core**, either $\sigma(e_1)$ or $\sigma(e_2)$, and either $\sigma(e_4)$ or $\sigma(e_5)$ are not insertion-only. Similarly, for such a **hook core**, either $\sigma(e_2)$ or $\sigma(e_3)$ is not insertion-only.

(Path core). We first consider the path core of five relations $e_1, e_2, e_3, e_4, e_5 \in \mathcal{E}$ and four attributes $x_1, x_2, x_3, x_4 \in \mathcal{V}$ with $x_1 \in e_1 \cap e_2, x_2 \in e_2 \cap e_3, x_3 \in e_3 \cap e_4, x_4 \in e_4 \cap e_5$, such that there exists no relation $e \in \mathcal{E}$ containing both x_i, x_j with $|j - i| > 1$. We identify a relation $e_1' \in \mathcal{E}$ as follows:

- If e_1 is reducible, then e_1 is identified as e_1';
- If there exists a leaf node $e \in \mathcal{E}$ with $\{e_2, e_3, e_4, e_5\} \cap \text{path}(e_1, e) \neq \emptyset$, then e is identified as e_1';
- Otherwise, the root r of $\mathcal{T}$ is identified as e_1'. For an arbitrary leaf node $e \in \mathcal{E}$, if $\{e_2, e_3, e_4, e_5\} \cap \text{path}(e_1, e) \neq \emptyset$, we observe that $\{e_2, e_3, e_4, e_5\} \subseteq \text{path}(e_1, e)$. Suppose not, $\{e_2, e_3, e_4, e_5\} \nsubseteq \text{path}(e_1, e)$. There must exist some $i \in \{2, 3, 4, 5\}$ such that $e_i \in \text{path}(e_1, e)$ but $e_{i+1} \notin \text{path}(e_1, e)$ or $e_{i-1} \notin \text{path}(e_1, e)$. Wlog, assume $e_{i+1} \notin \text{path}(e_1, e)$. Then, $e_1 \in \text{path}(e_i, e_{i+1})$, violating the connect property of x_i. As $\{e_2, e_3, e_4, e_5\} \subseteq \text{path}(e_1, e)$ for every leaf node $e \in \mathcal{E}$, then e must only have one child, as well as the root r. Hence, r is also a reducible relation with its unique child as the anchor. Moreover, $\{e_2, e_3, e_4, e_5\} \cap \text{path}(e_1, r) = \emptyset$.

Note that e_1' is reducible and $\{e_2, e_3, e_4, e_5\} \cap \text{path}(e_1, e_1') = \emptyset$. We similarly find a reducible relation $e_5' \in \mathcal{E}$ such that $\{e_1, e_2, e_3, e_4\} \cap \text{path}(e_5, e_5') = \emptyset$. Moreover, $\text{path}(e_1, e_1') \cap \text{path}(e_5, e_5') = \emptyset$. Let $I \subseteq \mathcal{V}$ be the set of join attributes appearing in any pair of relations in $\text{path}(e_1, e_1')$, and let $J \subseteq \mathcal{V}$

be the set of join attributes appearing in any pair of relations in $\mathsf{path}(e_5, e_5')$. For every attribute $x \in \mathcal{V} - I - J - \{x_1, x_2, x_3, x_4\}$, we set a special value $\{*\}$ in its domain.

After identifying the path core and e_1', e_5', we can use Q to simulate Q_{path}. If Q can be maintained in $O(1)$, then we can maintain Q_{path} under $\sigma(\{e_1, e_2, e_3, e_4, e_5\})$, which contradicts to Example 4.6.

(Hook core). We consider the hook core of three relations e_1, e_2, e_3 and three distinct attributes $x_1 \in \mathbf{y}, x_2, x_3 \in \bar{\mathbf{y}}$, with $x_1 \in e_1 - e_2 - e_3, x_2 \in e_1 \cap e_2 - e_3, x_3 \in e_2 \cap e_3 - e_1$ such that there exists no relation $e \in \mathcal{E}$ both x_1 and x_3. Note that e_3 must be a descendent of e_1. Suppose not, we consider e' as the lowest common ancestor of e_1 and e_3. Note that $x_2 \in e$ or $x_3 \in e$. It could be the case that $e' = e_3$. Let e be the child of e' that is also an ancestor of e_1. If $x_2 \in e', x_2 \in e$; otherwise, the connect property of x_2 is not preserved in $\mathcal{T}$. Similarly, if $x_3 \in e', x_3 \in e$; otherwise, the connect property of x_3 is not preserved in $\mathcal{T}$. As $e \cap e' \cap \bar{\mathbf{y}} \neq \emptyset$, $e \notin \mathcal{E}_{\mathsf{con}}$. Hence, no relation in $\mathcal{E}_{\mathsf{con}}$ contains x_1, coming to a contradiction. We identify a relation $e_3' \in \mathcal{E}$ as follows:

- If e_3 is reducible, then e_3 is identified as e_3';
- If there exists a leaf node $e \in \mathcal{E}$ with $\{e_1, e_2\} \cap \mathsf{path}(e_3, e) \neq \emptyset$, then e is identified as e_3'.

Let $I \subseteq \mathcal{V}$ be the set of join attributes appearing in any pair of relations in $\mathsf{path}(e_3, e_3')$,

After identifying the hook core and e_3', we can use Q to simulate Q_{hook}. If Q can be maintained in $O(1)$, then we can maintain Q_{hook} under $\sigma(\{e_1, e_2, e_3\})$, which contradicts to Example 4.5. □

Proof of Lemma 4.16. We start with the case when the height of $\mathcal{T}$ is 2. By Definition 2.7, for any leaf node $e \in \mathcal{T}$ and tuple $t \in R_e$, $\lambda_{\mathcal{T}}(t) = 1$. Conversely, for any non-leaf node $e \in \mathcal{T}$, for any $t \in R_e$, given that $\mathcal{T}$ is compatible, then for any child node e' of e on $\mathcal{T}$, if $\sigma(\{e'\})$ is insertion-only or $\sigma(\{e, e'\})$ is FIFO, we can select only one disjoint effective lifespan in $\dot{I}(t)$, making $\lambda_{\mathcal{T}}(t)$ also equal to 1 for all non-child nodes of $\mathcal{T}$. Therefore, the enclosureness $\lambda_{\mathcal{T}} = 1$ for all σ-sequences. If the height of $\mathcal{T}$ is $\geqslant 2$, by Definition 2.7, $\lambda_{\mathcal{T}}$ may be larger than $O(1)$ on some σ-sequence S. Let $P := \langle e_1, e_2, \cdots, e_n \rangle$ be a path on $\mathcal{T}$, such that: (1) $e_i \in \mathcal{E} : i \in [n]$; (2) e_i is the parent node of e_{i+1} on $\mathcal{T}$ for all $1 \leqslant i < n$; (3) e_1 is the root of $\mathcal{T}$, or $p(e_1)$ is a generalized relation; (4) e_n is a leaf node.

Given any path P on $\mathcal{T}$ with $n \geqslant 2$, only $\sigma(\{e_2\})$ can be insertion-only, as if e_i, e_j exist that are both insertion-only, one node can be removed because the reducible conditions are transitive. Meanwhile, for $e_i : i \in [3, n]$, $\sigma(\{e_{i-1}, e_i\})$ is FIFO , as the requirement of a compatible join tree.

Lemma B.2. *For any relation e, let $Q := \pi_x R_e$ for some attribute $x \subseteq e$ and let t be a result of Q with lifespan $[t^+, t^-]$. For any $t' \in R_e$ such that $\pi_x t' = t$ and $t'^+ \geqslant t^+$, then $t^- \geqslant t'^-$.*

The correctness follows that the projection operator only extends the lifespan of any input tuple.

First, we assume that $\sigma(\{e_2\})$ is not insertion-only. For any $t_1 \in e_1$ with a lifespan of $[t_1^+, t_1^-]$, there can be at most one tuple t_i in $e_i - \mathcal{V}_\bullet$ for any $i \in [2, n]$ that can join with t_1, as $e_i - \mathcal{V}_\bullet$ is a subset of e_1. Let's denote $t_i \in e_i - \mathcal{V}_\bullet$ as the tuple that can join with t_1. The tuple t_1 belongs to $V_s(R_1)$ only if $t_2, \cdots, t_n$ exist in $e_2 - \mathcal{V}_\bullet, \cdots, e_n - \mathcal{V}_\bullet$ and are not in $V_s(R_1)$ if one or more t_i missed from $e_i - \mathcal{V}_\bullet$. The tuple t_2 can be inserted into or deleted from $\pi_{e_2 - \mathcal{V}_\bullet}$ multiple times, resulting in multiple disjoint lifespans. However, there cannot be a lifespan for t_2 such that $t_2^+ \geqslant t_1^+$ and $t_2^- \leqslant t_1^-$. According to Lemma B.2 and the fact that $\sigma(\{e_1, e_2\})$ is FIFO. Moreover, within the time interval $[t_2^+, t_1^-]$, there cannot exist two disjoint lifespans for t_3 that are fully covered by the time interval. Otherwise, we could find three tuples t_1, t_2, t_3 from R_1, R_2, R_3, such that $t_1^+ \leqslant t_2^+ \leqslant t_3^+$, and $t_1^- \geqslant t_3^-$, but it must be $t_1^- \leqslant t_2^-$ and $t_2^- \leqslant t_3^-$ to ensure the FIFO property on both $\sigma(\{e_1, e_2\})$ and $\sigma(\{e_2, e_3\})$, which cannot exist.

Following the same idea, we can ascertain that there cannot exist any disjoint lifespans created by $t_i \in e_i - \mathcal{V}_\bullet$ within the lifespan $[t_1^+, t_1^-]$. Consequently, the status of t_1 can only change $O(1)$ times in a σ-sequence. A similar conclusion can be made for each $t_i \in e_i$, by considering the time interval $[t_i^+, t_i^-]$ resulting in $\lambda_{\mathcal{T}} = O(1)$ for any σ-sequence.

On the other hand, if $\sigma(\{e_2\})$ is insertion-only and let's consider t_2 as the first insertion on R_2 with insertion time t_2^+, then for any tuple $t_i \in e_i, i \in [3, n]$ such that t_i is inserted after time t_2^+, t_i won't be deleted from e_i, otherwise the FIFO property breaks. Additionally, the status of t_1 won't change until a corresponding t_2 is inserted. Therefore, once t_1 is inserted into $V_s(R_1)$, it will never be deleted from $V_s(R_1)$ until t_1 itself is removed from R_1. This results in $\lambda_{\mathcal{T}}(t) = O(1)$ for all $t \in e_1$. Similarly, as there won't be any deletion in e_j for the tuples that are inserted after t_i with any $j > i$, the enclosureness of any $t \in e_i$ will be 1, making the total enclosureness for $\lambda_{\mathcal{T}} = O(1)$. □

C DELETION-ONLY UPDATE SEQUENCES

In [25], it is observed that for any insertion-only update sequence S, we can construct a deletion-only update sequence S' symmetrically: (1) initializing the database as D_∞ at negative timestamp with an arbitrary order, where D_∞ is the database at timestamp $+\infty$ defined by S; and (2) changing every update $u = (t, s, +, R_i)$ to $u' = (t, -s, -, R_i)$. Note that S' has the same enclosureness as S, so it can be maintained with the same cost as S using the algorithm from [25]. Below, we focus on the hardness of deletion-only update sequences.

Theorem C.1. *For any $\epsilon > 0$, there is no data structure for $Q_7 := \pi_{x_1,x_3}(R_1(x_1, x_2) \bowtie R_2(x_2, x_3))$ that can be updated in $O(1)$ time while supporting $O(1)$-delay enumeration over any deletion-only update sequences, assuming the BMM conjecture.*

Proof. Given an instance of BMM $M \times M'$, where the two matrices M, M' has size $n \times n$, we can create the following deletion-only update sequence on Q_6 as follows: (1) we insert all tuples $(i, j) \in [n] \times [n]$ into R_1 and R_2 at negative timestamps; (2) for any (i, j) where $M_{ij} = 0$, we delete (i, j) from R_1; similarly, for any (i, j) where $M'_{ij} = 0$, we delete (i, j) from R_2; (3) after all deletions are done, we enumerate the query result. It is clear that if this CQ can be updated in $O(1)$ time while supporting $O(1)$-delay enumeration over any deletion-only update sequence, then by the BMM problem in time at most $O(n^2 \times 1 + n^2 \times 1) = O(n^2)$, which contradicts to BMM conjecture. □

Theorem C.2. *For any $\epsilon > 0$, there is no data structure for $Q_8 := R_1(x_1, x_2) \bowtie R_2(x_2, x_3) \bowtie R_3(x_3, x_1)$ that can be updated in $O(1)$ time while supporting $O(1)$-delay enumeration over any deletion-only update sequences, assuming HC conjecture.*

Proof. Given an instance of $(3, 2)$-HC problem, a special case of $(k + 1, k)$-HC problem, which tries to find all triangles from a graph. A conjecture is derived from the $(k + 1, k)$-HC conjecture on this special problem: We can create the following deletion-only update sequence on Q_8 to simulate the triangle problem: (1) we initial R_1, R_2 and R_3 with all tuples $(i, j) \in [n] \times [n]$; (2) if $(i, j) \notin G$, we delete (i, j) from R_1, R_2, R_3; (3) after all the absent entries are deleted, we enumerate the query result. If any results exist, then a triangle is detected from G. For the construction above, there are at most $O(n^2)$ updates, making the total running time of $O(n^2)$ if the query can be maintained in $O(1)$ time, which is a contradiction to the triangle conjecture as well as the HC conjecture. □

Theorem C.3. *For any non-free-connex CQ, no index can be updated in $O(1)$ amortized time while supporting $O(1)$-delay enumeration over any deletion-only update sequence, assuming the Boolean Matrix Multiplication, Triangle Detection, and HyperClique conjectures.*

REFERENCES

[1] TPC-H Benchmark. http://www.tpc.org/tpch/

[2] Amir Abboud and Virginia Vassilevska Williams. 2014. Popular conjectures imply strong lower bounds for dynamic problems. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE, 434–443.

[3] Mahmoud Abo Khamis, Ahmet Kara, Dan Olteanu, and Dan Suciu. 2024. Insert-Only versus Insert-Delete in Dynamic Query Evaluation. *Proceedings of the ACM on Management of Data* 2, 5 (2024), 1–26.

[4] Yanif Ahmad, Oliver Kennedy, Christoph Koch, and Milos Nikolic. 2012. DBToaster: Higher-order delta processing for dynamic, frequently fresh views. *Proceedings of the VLDB Endowment* 5, 10 (2012), 968–979.

[5] Guillaume Bagan, Arnaud Durand, and Etienne Grandjean. 2007. On Acyclic Conjunctive Queries and Constant Delay Enumeration. In *Computer Science Logic*. Springer Berlin Heidelberg, Berlin, Heidelberg, 208–222.

[6] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. 1983. On the desirability of acyclic database schemes. *JACM* 30, 3 (1983), 479–513.

[7] Christoph Berkholz, Jens Keppeler, and Nicole Schweikardt. 2017. Answering Conjunctive Queries under Updates. In *PODS*. 303–318.

[8] Christoph Berkholz, Jens Keppeler, and Nicole Schweikardt. 2018. Answering UCQs under Updates and in the Presence of Integrity Constraints. In *ICDT*.

[9] Johann Brault-Baron. 2013. *On the relevance of énumération: complexitye in propositional and first order logics*. Ph. D. Dissertation. University of Caen.

[10] Nofar Carmeli and Luc Segoufin. 2023. Conjunctive Queries With Self-Joins, Towards a Fine-Grained Enumeration Complexity Analysis. In *PODS*. 277–289.

[11] Rada Chirkova and Jun Yang. 2012. Materialized views. *Foundations and Trends® in Databases* 4, 4 (2012), 295–405.

[12] R. Fagin. 1983. Degrees of acyclicity for hypergraphs and relational database schemes. *JACM* 30, 3 (1983), 514–550.

[13] Ashish Gupta, Inderpal Singh Mumick, and Venkatramanan Siva Subrahmanian. 1993. Maintaining views incrementally. In *ACM SIGMOD Record*, Vol. 22. ACM, 157–166.

[14] Kathrin Hanauer, Monika Henzinger, and Qi Cheng Hua. 2022. Fully Dynamic Four-Vertex Subgraph Counting. In *1st Symposium on Algorithmic Foundations of Dynamic Networks*.

[15] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. 2015. Unifying and Strengthening Hardness for Dynamic Problems via the Online Matrix-Vector Multiplication Conjecture. In *STOC*. 21–30.

[16] Muhammad Idris, Martin Ugarte, and Stijn Vansummeren. 2017. The Dynamic Yannakakis Algorithm: Compact and Efficient Query Processing Under Updates. In *SIGMOD*. 1259–1274.

[17] Ahmet Kara, Zheng Luo, Milos Nikolic, Dan Olteanu, and Haozhe Zhang. 2024. Tractable conjunctive queries over static and dynamic relations. In *28th International Conference on Database Theory*.

[18] Ahmet Kara, Hung Q. Ngo, Milos Nikolic, Dan Olteanu, and Haozhe Zhang. 2020. Maintaining Triangle Queries under Updates. *ACM Trans. Database Syst.* 45, 3, Article 11 (aug 2020), 46 pages.

[19] Ahmet Kara, Milos Nikolic, Dan Olteanu, and Haozhe Zhang. 2020. Trade-offs in static and dynamic evaluation of hierarchical queries. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 375–392.

[20] Christoph Koch. 2010. Incremental query evaluation in a ring of databases. In *Proc. ACM SIGMOD International Conference on Management of Data*. ACM, 87–98.

[21] Andrea Lincoln, Virginia Vassilevska Williams, and Ryan Williams. 2018. Tight hardness for shortest cycles and paths in sparse graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 1236–1252.

[22] Milos Nikolic, Mohammad Dashti, and Christoph Koch. 2016. How to win a hot dog eating contest: Distributed incremental view maintenance with batch updates. In *Proc. ACM SIGMOD International Conference on Management of Data*. ACM, 511–526.

[23] Milos Nikolic and Dan Olteanu. 2018. Incremental view maintenance with triple lock factorization benefits. In *Proc. ACM SIGMOD International Conference on Management of Data*. ACM, 365–380.

[24] Amir Shpilka. 2003. Lower bounds for matrix product. *SIAM J. Comput.* 32, 5 (2003), 1185–1200.

[25] Qichen Wang, Xiao Hu, Binyang Dai, and Ke Yi. 2023. Change Propagation Without Joins. *Proceedings of the VLDB Endowment* 16, 5 (2023), 1046–1058.

[26] Qichen Wang and Ke Yi. 2020. Maintaining Acyclic Foreign-Key Joins under Updates. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1225–1239.