# Answering Aggregate Queries with Ordered Direct Access

Idan Eldar[1], Nofar Carmeli[2] and Benny Kimelfeld[1]

[1]Technion – Israel Institute of Technology, Haifa 3200023, Israel
[2]Inria, LIRMM, Univ Montpellier, CNRS, France

Abstract

The paper presents recent findings on the fine-grained complexity of conjunctive queries with aggregation. For common aggregate functions (e.g., min, max, sum), such a query can be phrased as an ordinary conjunctive query over a database annotated with a suitable commutative semiring. We investigate the ability to evaluate queries by constructing in log-linear time a data structure that provides logarithmic-time direct access to the answers, ordered by a lexicographic order of choice. Importantly, these complexity guarantees hold even if the result of the query is asymptotically larger than log-linear in the size of the input.

## 1. Direct Access to Answers of Conjunctive Queries

Consider a query $Q$ that may have a large number of answers, say cubic in the number of tuples of the input database $D$. By answering $Q$ via *direct access*, we avoid the materialization of the list of answers, and instead, construct a compact data structure $S$ that supports random access: given an index $i$, retrieve the $i$th answer. Hence, direct-access evaluation for a query $Q$ consists of two algorithms: the *preprocessing* algorithm constructs a data structure $S_D$ from an input database $D$, and the *access* algorithm takes as input $S_D$ and an index $i$, and returns the $i$th answer. If $i$ is greater than the total number of answers, then the algorithm returns *null*. We say that the evaluation of $Q$ is in $\langle \text{loglinear}, \log \rangle$ if such algorithms exist so that preprocessing takes log-linear time (that is, linear possibly multiplied by logarithmic factors) and access returns an answer in logarithmic time. Note that $S$ may be considerably cheaper to construct than $Q(D)$.

Direct-access solutions have been initially devised for Conjunctive Queries (CQs) as a way to establish algorithms for enumerating the answers with linear preprocessing time and constant delay [1]. Later, direct access played a crucial role within the task of enumerating the answers in a uniformly random order [2]. When direct access got recognized as a goal in its own right [3], the natural next step was to ask which *orders* over the answers allow for such evaluation.

The following example is inspired by the FIFA World Cup. We have a database of players of teams (countries), sponsors, games, and goals. Specifically, we have three relations: Teams(*player*, *country*), Sponsors(*org*, *country*), and Goals(*game*, *player*, *time*). The following

*AMW'23: 15th Alberto Mendelzon International Workshop on Foundations of Data Management, May 22–26, 2023, Santiago, Chile*

idel@campus.technion.ac.il (I. Eldar); nofar.carmeli@inria.fr (N. Carmeli); bennyk@technion.ac.il (B. Kimelfeld)
0000-0003-0673-5510 (N. Carmeli); 0000-0002-7156-1572 (B. Kimelfeld)



CEUR Workshop Proceedings (CEUR-WS.org)

CQ finds times where sponsors got exposure due to goals of supported teams:

$$Q_1(c, o, p, t) \text{ :- } \textsc{Teams}(p, c), \textsc{Sponsors}(o, c), \textsc{Goals}(g, p, t)$$

Suppose also that we would like the answers to be ordered lexicographically by their order in the head: first by country, then by organization, and so on. Note that $c$, $o$, $p$ and $t$ are *free* variables and $g$ is an *existential* variable. Carmeli et al. [4] studied the ability to evaluate such ordered queries with direct access. In the case of $Q_1$, they show that there is an efficient direct-access evaluation, since the query is *free-connex* and there is no *disruptive trio*. We explain these terms next. (For more precise definition, we refer the reader to Carmeli et al. [4].)

In general, a CQ has the form $Q(\vec{x}) \text{ :- } \varphi_1(\vec{x}, \vec{y}), \ldots, \varphi_\ell(\vec{x}, \vec{y})$ where $\vec{x}$ and $\vec{y}$ are disjoint sequences of variables, and each $\varphi_i(\vec{x}, \vec{y})$ is an *atomic query*. A CQ is free-connex if it is acyclic and remains acyclic even if the head is viewed as an atom. Also recall that a *disruptive trio* of a CQ $Q(\vec{x})$ is a set of three distinct free variables $x_1$, $x_2$, and $x_3$ such that $x_1$ and $x_2$ neighbor $x_3$ but not each other, and $x_3$ appears after both $x_1$ and $x_2$ in $\vec{x}$. Two variables are considered neighbors if they appear together in some query atom.

## 2. Ordered CQs over Annotated Databases

In this work, we continue with the line of work of Carmeli et al. [4] and investigate the ability to support query evaluation via direct access for queries over annotated databases. To illustrate, suppose that we would like to count the goals per sponsorship and player. We can phrase this query as follows.

$$Q_2(c, \mathsf{Count}(), o, p) \text{ :- } \textsc{Teams}(p, c), \textsc{Sponsors}(o, c), \textsc{Goals}(g, p, t)$$

Here, we order the answers first by $c$, then by Count(), and then by $o$ and $p$. Semantically, $p$, $c$, and $o$ are treated as the *grouping variables* (rather than free variables), where each combination of values defines a group of tuples over $(p, c, o, g, t)$ and Count() counts the tuples in the group.

CQs with some common aggregate functions can be translated into ordinary CQs over annotated databases [5, 6]. We adopt the well-known framework of *provenance semiring* [7] and phrase the query as an ordinary CQ with the annotation carrying the aggregate value (number of goals in our example). To design direct-access evaluation, we found it more elegant, general, and insightful to support annotation rather than SQL-like aggregate functions. For illustration, we can phrase $Q_2$ as the following query.

$$Q_3(c, \star, o, p) \text{ :- } \textsc{Teams}(p, c), \textsc{Sponsors}(o, c), \textsc{Goals}(g, p, t)$$

Ignoring the $\star$ symbol, this is an ordinary CQ applied when the database is annotated by elements from the domain $\mathbb{K}$ of a commutative semiring $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$. In a nutshell, the idea is that each tuple is annotated with an element of the semiring, the annotation of each tuple in the group is the product of the participating tuple annotations, and the annotation of the whole group is the sum of all tuple annotations in the group's tuples. We can use different semirings and annotations to compute different aggregate functions like sum, min, and max. In the case of $Q_3$, for instance, the semiring is $(\mathbb{Z}, +, \cdot, 0, 1)$ and each tuple is annotated simply with the

number 1. Moreover, we order by $c$, then by the annotation, and then by $o$ and $p$. Notationally, we specify the annotation position by the symbol $\star$ and refer to the query as a CQ$^\star$, which is defined similarly to a CQ but with $\star$ added to the head. More precisely, we write a CQ$^\star$ as $Q(\vec{x}, \star, \vec{z})$ to denote that $\star$ is between the (possibly empty) sequences $\vec{x}$ and $\vec{z}$ of head variables.

## 3. Complexity Results

We now describe our results on direct access for CQ$^\star$s. As conventionally done in fine-grained analysis of queries, we use the RAM model [8]. We assume logarithmic space for representing each element of the semiring, and constant time for the operations $\oplus$ and $\otimes$. We also assume that the semiring domain is ordered, and comparison is in constant time.

We begin with the case where the order *does not* involve the annotation. Equivalently, we discuss CQ$^\star$s of the form $Q(\vec{x}, \star)$. The following theorem states that, in this case, the results of Carmeli et al. [4] continue to hold for annotated databases. The theorem refers to the HYPERCLIQUE and SparseBMM hypotheses, which are common assumptions of lower bounds in fine-grained complexity. (We refer the reader to Carmeli et al. [4] for details.)

**Theorem 1.** *Let $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ be a commutative semiring and $Q(\vec{x}, \star)$ a CQ$^\star$.*

1. *If $Q$ is free-connex and with no disruptive trio, then direct access for $Q$ is in $\langle \text{loglinear}, \log \rangle$ on $\mathbb{K}$-databases.*
2. *Otherwise, if $Q$ is also self-join-free, then direct access for $Q$ is not in $\langle \text{loglinear}, \log \rangle$, assuming the HYPERCLIQUE hypothesis (in case $Q$ is cyclic) and the SparseBMM hypothesis (in case $Q$ is acyclic).*

The following theorem states a lower bound in an extremely simple query (Cartesian product) for the semirings used to compute the aggregates count, sum, min, and max, assuming the 3SUM conjecture [9, 10].

**Theorem 2.** *Let $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ be one of the counting, numerical, max tropical, or min tropical semirings. Direct access for the CQ$^\star$ $Q(\star, x, y) :- R(x), S(y)$ is not in $\langle \text{loglinear}, \log \rangle$ over $\mathbb{K}$-databases, assuming the 3SUM conjecture.*

Theorem 2 implies that, to obtain evaluation algorithms in $\langle \text{loglinear}, \log \rangle$ while incorporating the annotation in the order, we need to restrict the class of CQ$^\star$s or make assumptions on the annotated database. In the following theorem, we restrict the structure of the CQ$^\star$. We also make the mild assumption that the semiring is $\otimes$-*monotone*, that is, the function $f_c$ is monotone for every $c \in \mathbb{K}$, where $f_c : \mathbb{K} \to \mathbb{K}$ is defined by $f_c(y) = c \otimes y$.

**Theorem 3.** *Let $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ be a $\otimes$-monotone commutative semiring, and $Q(\vec{x}, \star, \vec{z})$ a free-connex CQ$^\star$ with no disruptive trio. If every atom of $Q$ contains either all variables of $\vec{z}$ or none of them, then direct access for $Q$ is in $\langle \text{loglinear}, \log \rangle$.*

As an example, the CQ$^\star$ $Q(w, x, \star, y, z) :- R(w, x), S(x, y, z), T(y, z)$ is in $\langle \text{loglinear}, \log \rangle$ over databases annotated with the numerical semiring.

The final result that we explain in this section holds under two assumptions that we explain next. For the first assumption, consider the aggregate query

$$Q(\mathsf{Max}(w), x, y) :- R(x, w), S(y)$$

When translating $Q$ into a CQ$^\star$, we obtain $Q(\star, x, y) :- R(x), S(y)$ over $\mathbb{Q}$-databases annotated by the max tropical semiring $(\mathbb{Q}\cup\{-\infty\}, \mathsf{max}, +, -\infty, 0)$. Hence, according to Theorem 2, we translate the problem into an intractable one. Nevertheless, direct access for $Q$ by $(\star, x, y)$ is, in fact, in $\langle \text{loglinear}, \log\rangle$. This discrepancy stems from the fact that the hardness in Theorem 2 relies on the annotation of tuples from both $R$ and $S$. Yet, in our translation, all $S$-facts are annotated by 1. The resulting $\mathbb{K}$-database is such that every fact is annotated by $\bar{1}$ (the multiplicative identity), with the exception of one relation. We call such a $\mathbb{K}$-database *$R$-annotated*.

The second assumption is that the operation $\oplus$ is *idempotent*, that is, for every $a$ in the domain $\mathbb{K}$ we have that $a \oplus a = a$. This is the case when we start with the aggregate functions min and max, for example.

The following theorem states that, under the above assumptions, we can effectively classify every CQ$^\star$ without self-joins into two categories:

1. CQ$^\star$s with direct access in $\langle \text{loglinear}, \log\rangle$;
2. CQ$^\star$s where direct access is not in $\langle \text{loglinear}, \log\rangle$ under standard complexity assumptions and under the assumption that the domain $\mathbb{K}$ contains the domain of natural numbers. (In fact, it suffices that we can generate an infinite increasing sequence of elements in $\mathbb{K}$.)

This is done by converting the CQ$^\star$ $Q$ into an ordinary CQ $Q_0$, so that direct access for the two is computationally equivalent. We can then use previous results [4] to determine the feasibility of efficient direct access.

**Theorem 4.** *Let $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ be a $\oplus$-idempotent commutative semiring. There exists a polynomial-time algorithm that takes as input a free-connex CQ$^\star$ $Q$ without self-joins and a relation symbol $R$ of $Q$, and produces a full acyclic CQ $Q_0$ without self-joins, so that the following hold.*

1. *If $Q_0$ has no disruptive trio, then direct access for $Q$ over $R$-annotated $\mathbb{K}$-databases is in $\langle \text{loglinear}, \log\rangle$.*
2. *Otherwise, in the case where $\mathbb{N} \subseteq \mathbb{K}$, direct access for $Q$ over $R$-annotated $\mathbb{K}$-databases is not in $\langle \text{loglinear}, \log\rangle$, assuming the SparseBMM hypothesis.*

As an example, consider the following CQ$^\star$:

$$Q(\star, x_1, x_2, x_3) :- R(x_1, x_3, w_3), S(x_2, x_3), T(x_3, w_1), U(w_1, w_2)$$

The translation of theorem 4 reduces direct access for $Q$ on $U$-annotated $\mathbb{K}$-databases to direct access for the following CQ:

$$Q_0(y, x_1, x_2, x_3) :- R'(x_1, x_3, y), S'(x_2, x_3, y), T'(x_3, y)$$

Since $Q_0$ does not have a disruptive trio, direct access for $Q_0$ is in $\langle \text{loglinear}, \log\rangle$, and therefore, so is direct access for $Q$ on $U$-annotated $\mathbb{K}$-databases.

See the full version of this paper [11] for more details about the results described here.

### Acknowledgments

The work of Idan Eldar and Benny Kimelfeld was supported by the Israel Science Foundation (ISF), Grant 768/19, and the German Research Foundation (DFG) Project 412400621 (DIP program).

## References

[1] J. Brault-Baron, De la pertinence de l'énumération : complexité en logiques propositionnelle et du premier ordre, Theses, Université de Caen, 2013.

[2] N. Carmeli, S. Zeevi, C. Berkholz, A. Conte, B. Kimelfeld, N. Schweikardt, Answering (unions of) conjunctive queries using random access and random-order enumeration, ACM Trans. Database Syst. 47 (2022) 9:1–9:49.

[3] K. Bringmann, N. Carmeli, S. Mengel, Tight fine-grained bounds for direct access on join queries, in: PODS, ACM, 2022, pp. 427–436.

[4] N. Carmeli, N. Tziavelis, W. Gatterbauer, B. Kimelfeld, M. Riedewald, Tractable orders for direct access to ranked answers of conjunctive queries, in: PODS, ACM, 2021, pp. 325–341.

[5] C. Ré, D. Suciu, The trichotomy of HAVING queries on a probabilistic database, VLDB J. 18 (2009) 1091–1116.

[6] M. A. Khamis, R. R. Curtin, B. Moseley, H. Q. Ngo, X. Nguyen, D. Olteanu, M. Schleich, Functional aggregate queries with additive inequalities, ACM Trans. Database Syst. 45 (2020) 17:1–17:41.

[7] T. J. Green, G. Karvounarakis, V. Tannen, Provenance semirings, in: PODS, PODS '07, Association for Computing Machinery, New York, NY, USA, 2007, p. 31–40.

[8] Étienne Grandjean, L. Jachiet, Which arithmetic operations can be performed in constant time in the ram model with addition?, 2023. arXiv:2206.13851.

[9] A. Gajentaan, M. H. Overmars, On a class of o(n2) problems in computational geometry, Computational Geometry 5 (1995) 165–185.

[10] M. Patrascu, Towards polynomial lower bounds for dynamic problems, in: Proceedings of the Forty-Second ACM Symposium on Theory of Computing, STOC '10, Association for Computing Machinery, New York, NY, USA, 2010, p. 603–610.

[11] I. Eldar, N. Carmeli, B. Kimelfeld, Direct access for answers to conjunctive queries with aggregation, 2023. arXiv:2303.05327.