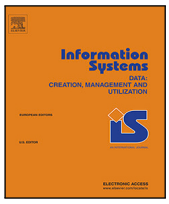
Information Systems 135 (2026) 102583

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is



Temporal relational algebras supporting preferences in temporal relational databases: Definition, properties and evaluation

Luca Anselma [a,*], Antonella Coviello [b], Davide Cerotti [c], Erica Raina [c], Paolo Terenziani [c]

[a] Dipartimento di Informatica, Università degli Studi di Torino, Torino, Italy
[b] Politecnico di Torino, Torino, Italy
[c] DISIT, Università del Piemonte Orientale, Alessandria, Italy

ARTICLE INFO

Keywords:
Temporal relational databases
Temporal relational algebra
Temporal indeterminacy
Preferences

ABSTRACT

Despite numerous approaches address the treatment of time within relational contexts, temporal preferences remain unexplored. Many tasks and applications, such as planning, scheduling, workflows, and guidelines, involve scenarios where the exact timing of events is not known — referred to as *indeterminate time*. In such cases, preferences can be assigned to different possible temporal outcomes. In a recent study, we established the theoretical foundation for handling preferential indeterminate time in temporal relational databases. This includes proposing a temporal relational representation and a corresponding temporal relational algebra, along with an analysis of their theoretical properties, such as correctness and reducibility.

The contributions of this paper are twofold. First, we extend the above theoretical framework to deal with a more expressive representation of temporal preferences. Second, we assess both theoretical frameworks in terms of performance evaluation along different dimensions, and study the overhead added to cope with preferences with respect to relational approaches without time, with exact time, and with indeterminate time but no preferences.

1. Introduction

Most phenomena and data inherently involve a temporal dimension, necessitating databases (DBs) to address this aspect effectively. Since the 1990s, researchers have recognized the unique characteristics of temporal data, advocating for its management through *specialized* techniques [1]. For example, Section 1 of Snodgrass et al.'s seminal book [1] provides insightful examples illustrating the challenges of handling even seemingly straightforward queries — such as joins between relations or projections over a subset of attributes — when considering the *valid time* of tuples. These examples highlight the inherent complexity of temporal data management for both researchers and practitioners unfamiliar with the field.

Hundreds of *dedicated approaches* have been developed to address various temporal phenomena in *temporal relational databases* (hereafter referred to as TDBs) [2]. Among these, TSQL2 represents a consensus approach by the TDB research community and serves as a cornerstone in the field [1]. TSQL2 comprises a temporal data model and a query language, built upon an explicit semantics (BCDM) and a temporal algebra. Its theoretical foundations include the reducibility of TSQL2 algebra to Codd's relational algebra [3], ensuring compatibility with non-temporal database approaches, as detailed in [1]. Today, most major database management systems provide temporal support, and research in the TDB domain remains active, as highlighted by recent surveys [4] and the updated Encyclopedia of Database Systems [5].

Despite significant advancements, several challenges in TDB research remain unresolved. For example, most TDB approaches primarily focus on the *valid time* of facts — the time when events occur [1] — and/or the *transaction time*—the time of their insertion or deletion in the database [1]. Both are typically assumed to be precisely known. However, in many real-world scenarios and applications (e.g., workflows, guidelines, planning, scheduling), the exact valid time is not exactly known, a situation referred to as *temporal indeterminacy* [6].

Let us consider Ex.1.

Ex. 1. Mary's physical therapy must start between March 10 and March 30 and last 4–20 days. ■

* Corresponding author.
E-mail addresses: luca.anselma@unito.it (L. Anselma), antonella.coviello@studenti.polito.it (A. Coviello), davide.cerotti@uniupo.it (D. Cerotti), erica.raina@uniupo.it (E. Raina), paolo.terenziani@uniupo.it (P. Terenziani).

https://doi.org/10.1016/j.is.2025.102583
Received 2 January 2025; Received in revised form 12 June 2025; Accepted 2 July 2025
Available online 17 July 2025

Notably, Ex. 1 is an implicit way of denoting (if we consider the granularity of days) the $21 \cdot 17 = 357$ precise scenarios/instantiations in Ex. 1'.

Ex. 1'.
Mary's therapy starts on March 10 and lasts 4 days, or
Mary's therapy starts on March 10 and lasts 5 days, or
...
Mary's therapy starts on March 30 and lasts 20 days. ■

Although valid time is often only approximately known in many applications, few TDB approaches have addressed the challenge of *temporal indeterminacy*. Notable contributions in this area include [6–11]. However, these approaches do not account for the fact that, in many contexts, different scenarios or instantiations of a temporally indeterminate fact may not be equally preferable. As noted by Dubois et al. [12], it is often necessary to establish a "*ranking of the instantiations that are acceptable*".

Research in fields such as Artificial Intelligence underscores the importance of managing preferences in relation to temporal indeterminacy (e.g., to specify a degree of preference about the different possible times of occurrence of facts and events). As noted in [13], "**temporal preferences** are quantitative preferences that pertain to the position and duration of events in time".

Temporal preferences are crucial for **personalization** and play a significant role in various domains, including *economics* [14] and *medicine* [15,16]. Their inclusion allows for tailoring approaches to better address the specific needs and priorities of different applications and users. Though our approach is completely *domain-independent*, we have in mind a specific application of our general framework in the medical (and, more specifically, clinical guideline) context, within the AI-LEAP (Learning Personalization of AI and with AI) project [17,18].

In this context, it is often neither feasible nor practical to specify exact timings for actions [19]. Instead, temporal constraints within guidelines are generally interpreted as broad recommendations, which can be adhered to with varying degrees of medical preference [15]. In many cases, medical temporal constraints are expressed in terms of a range of time for the starting time of an action, and a range of possible durations, associated with their preferences (consider, for instance, Ex. 2).

Ex. 2. Mary's physical therapy must start between March 5 and March 13 and last for 6–8 days with high preference, start between March 4 and March 14 and last for 5–13 days with a medium preference, and start between March 1 and March 21 and last 4–20 days with a low preference. ■

In more complex situations, also a range of possible values for the ending time of the action has to be specified. Consider, e.g., Ex. 2'.

Ex. 2'. Mary's physical therapy must start between March 5 and March 13, end between March 11 and March 21, and last for 6–8 days with high preference, start between March 4 and March 14, end between March 9 and March 27 and last for 5–13 days with a medium preference, and start between March 1 and March 21, end between March 8 and March 30 and last 4–20 days with a low preference. ■

As we will discuss in Section 3, a formalism considering (i) a range of possible starting times, (ii) a range of possible ending times, and (iii) a range of possible durations is strictly more expressive than one considering only starting time and duration (or starting time and ending time). For instance, in Ex. 2', considering low preference, stating that the ending time ranges from March 8 to March 30 allows one to exclude the possibility that, when starting on March 1, the duration is 4–6 days.

In a recent paper, we have laid the theoretical basis of the first framework supporting preferential indeterminate time in TDBs [20], proposing an approach considering starting time, duration and preferences (consider, e.g., Ex. 2). We have proposed a relational data model coping with preferential indeterminate time, and a temporal relational algebra to query it, and we have studied the theoretical properties (e.g., data semantics, correctness, and reducibility of the algebra).

In this paper, we propose two main original contributions:

(1) We propose a new theoretical framework, extending the one in [20], to deal with preferential time considering indeterminacy about starting time, ending time and durations (see, e.g., Ex. 2'), and study its properties.
(2) We assess both the theoretical framework in [20] and the one at item (1) above in terms of performance evaluation. We analyze the performance of both frameworks along different dimensions and study the added overhead of dealing with preferences.

The paper is organized as follows. To make the paper self-contained, in Section 2 we briefly summarize the main results in [20], with specific attention to the aspects related to the performance analysis carried on in this paper. In Section 3 we propose a new relational framework coping with starting time, ending time, duration and preferences (see Contribution (1) above). In particular, we first propose a relational representation, and then we propose a temporal relational algebra to query it. Finally, we prove the correctness and the reducibility of such an algebra. To make our paper more readable for non-expert readers, some technical aspects of the definition of the algebraic operators and the proofs of correctness and reducibility are reported in Appendices (Appendix A and Appendix B respectively). Notably, we regard such appendices as integral and essential contributions of our work, granting its theoretical soundness.

In Section 4, we present an experimental evaluation comparing the framework introduced in [20] with the new approach proposed in Section 3. Specifically, we outline the implementation of the theoretical framework and evaluate the performance of the temporal algebraic operators across sample relations of varying dimensions and scales of preferences. We also benchmark our approach against a reference method, which directly manages the explicit semantics of the data model as detailed in Section 2. Additionally, we analyze the overhead introduced by incorporating temporal preferences in comparison to relational approaches that (i) exclude time, (ii) handle exact time, and (iii) manage indeterminate time without preferences. Section 5 discusses the related work in the literature, emphasizing the original contributions of this paper with respect to the state of the art (including our previous publications on this topic). Section 6 concludes with a discussion of findings and directions for future work.

2. Preliminaries

In [20], we introduced the first TDB model and relational algebra designed to address *temporal indeterminacy with preferences* for starting times and durations (see Ex. 2). Our methodology builds on the general framework proposed in [9] and includes the following contributions:

(1) A compact *data representation model*;
(2) A *relational algebra* for querying the model;
(3) Definition of the *data semantics*;
(4) Proof of the algebra's *correctness*;
(5) Proof of *reducibility* to Codd's standard relational algebra.

To make this paper self-contained, we revisit points (1) and (2) in Sections 2.1 and 2.2, respectively, while points (3), (4), and (5) are briefly summarized in Section 2.3.

2.1. Data model

As in TSQL2 [1], BCDM [21], and other works reviewed in [1], time in our model is discrete, linearly ordered, and isomorphic to a subset of integers. The fundamental unit of time is the *chronon*. We denote the chronon domain as T^C. In this study, we assume days as the unit of chronons.

Preferences are treated as qualitative (non-numeric), and our model supports any scale of preferences as long as the elements are totally ordered. For instance, in Ex. 2 the preference scale $\langle low, medium, high \rangle$ is used. We denote as S_r a scale of r ordered preference levels.

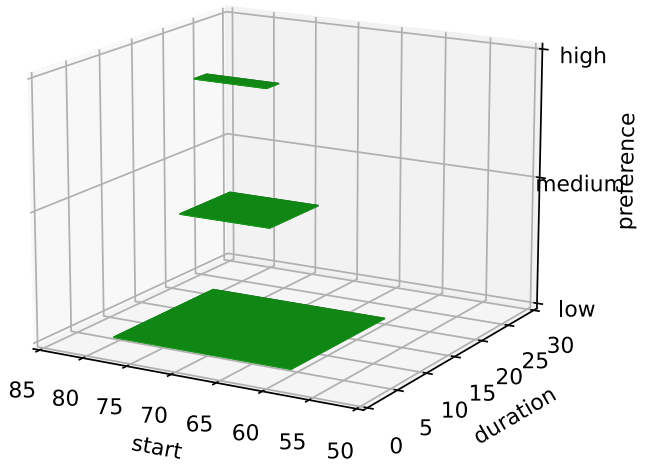


Fig. 1. Pyramid preferences for Ex. 2.

Each fact (event, property, or other temporal data) is associated with a *valid time* [1] and corresponding preferences. For each preference level (e.g., "low", "medium", "high"), the model specifies the temporal range with at least that level of preference using a pair of intervals $\langle [s_m, s_M), [d_m, d_M) \rangle$ such that $s_m, s_M \in T^C$, $s_m < s_M$, $d_m, d_M \in \mathbb{N}$, $d_m < d_M$. s_m represents the minimum starting chronon, s_M the maximum starting chronon, d_m the minimum duration, and d_M the maximum duration.

Our work in [22] focused on *layered preferences*, which are common in many applications. These functions represent preferences as "pyramids" centered around specific temporal values, where preferences decrease as one moves away from the center. This creates nested ranges with the highest preference at the top and progressively lower preferences in the layers below.

Fig. 1 illustrates such a pyramid of preferences for Ex. 2. In the example, we assume that time starts at the beginning of the year, and we adopt the granularity of days. Thus, March 5 corresponds to day 64, and March 13 to day 72. Here, the top rectangle (coordinates $(64, 73, 6, 9)$) indicates a high preference (*preference* axis) for starting between days 64 and 73 (*start* axis) with durations of 6 to 9 days (*duration* axis). The low-preference rectangle (coordinates $(60, 81, 4, 21)$) encompasses both the medium- and high-preference rectangles, reflecting the nested structure of the preference hierarchy.

A ***temporal relation with layered preferences*** associates facts with valid times with layered preferences. Given a scale S_r with r levels (which is unique for a given DB), the ***schema*** of a ***temporal relation with layered preferences*** $R = (A_1, \ldots, A_n | (T^1_{sm} : T^1_{sM}, T^1_{dm} : T^1_{dM})^1, \ldots, (T^r_{sm} : T^r_{sM}, T^r_{dm} : T^r_{dM})^r)$ consists of an arbitrary number of non-temporal attributes $A_1, \ldots, A_n$ encoding some fact, and of r quadruples $(T^i_{sm} : T^i_{sM}, T^i_{dm} : T^i_{dM})$ of temporal attributes to encode $\langle [s_m, s_M), [d_m, d_M) \rangle$

Ex. 3. Ex. 2 can be represented, using the scale S_3: $\langle low, medium, high \rangle$, by the relation pat_treatm^{TP} with schema $(patient\_t, treatment | (T^1_{sm} : T^1_{sM}, T^1_{dm} : T^1_{dM})^1, (T^2_{sm} : T^2_{sM}, T^2_{dm} : T^2_{dM})^2, (T^3_{sm} : T^3_{sM}, T^3_{dm} : T^3_{dM})^3)$ as following: $pat\_treatm^{TP} = \{(Mary, phys | (60 : 81, 4 : 21)^1, (63 : 74, 5 : 14)^2, (64 : 73, 6 : 9)^3)\}$. ■

2.2. Relational algebra

Most TDB approaches adopt the convention that temporal relational algebra operators extend standard Codd's operators. For non-temporal attributes, these operators behave identically to their traditional counterparts. For temporal attributes, however, they employ operations like *intersection* (for temporal Cartesian product $\times^T$) and *difference* (for temporal difference $-^T$)[1]. Temporal selection, projection, and union, on the other hand, typically align with Codd's standard definitions [1,5,23]. Our approach aligns with this widely accepted framework. While the formal definitions of our temporal algebraic operators — particularly temporal difference — are complex and detailed in [20], the underlying concept is intuitive and can be illustrated graphically. Specifically, *temporal Cartesian product* involves finding the intersection of two pyramids, *temporal difference* involves subtracting one pyramid from another. Fig. 2 provides an example: the temporal Cartesian product results in the intersection of two pyramids (the orange pyramid on the left), while the temporal difference results in the portions of the red pyramids that do not overlap with the black one (on the right). For the rest of this paper, we denote the operators from [20] with the superscript "TP" (Temporal Preference).

2.3. Data semantics, correctness and reducibility

Our approach provides a compact and implicit representation for associating valid times and preferences with facts. Each rectangle in the representation is summarized by its four vertices, which encapsulate all bi-temporal points within it (see Fig. 1).

Following the methodology in [11], we define the semantics of our data model using the function $MakeExplicit$. This function expands the compact representation into an explicit set of all temporal triples $\langle s, d, p \rangle$, where s is the start time, d is the duration, and p is the preference level.

Specifically, $MakeExplicit$, applied to a temporal relation in our approach, provides as output a new relation in which each fact is paired with the explicit set of all the triples $\langle s, d, p \rangle$ such that x, starting at s and lasting d, has at least preference p.

Ex. 4. $MakeExplicit(pat\_treatm) = \{(Mary, phys | \{\langle 60, 4, L \rangle, \ldots, \langle 80, 20, L \rangle, \langle 63, 5, M \rangle, \ldots, \langle 73, 13, M \rangle, \langle 64, 6, H \rangle, \ldots, \langle 72, 8, H \rangle\})\}$. For the sake of brevity, here we have used L, M, and H to denote *low*, *medium*, and *high* respectively. ■

Since our operators manipulate the compact representation directly, we must demonstrate their correctness. Informally: for every operator Op^{TP}, applying it to compact representations and then expanding the result via $MakeExplicit$ yields the same outcome as first expanding the compact representations and then applying the corresponding explicit operator Op^{Expl}.

Reducibility is a fundamental property of all TDB approaches. It ensures that new operators, designed to handle advanced phenomena, simplify to their basic forms when those phenomena are disregarded [1, 5]. In [20], we proved that our algebraic operators satisfy this property, preserving compatibility with standard relational algebra.

3. Data model and algebra for starting time, ending time and duration with temporal preferences

This Section constitutes the core of this paper. We first motivate the need for an extension to the approach in [20], and informally discuss the technical problems to be solved to achieve such an extension in a theoretically-grounded framework (Section 3.1). Then, we propose a new data model (Section 3.2), specify its semantics (Section 3.3), define an algebra to query the data model (Section 3.4), and formally study the properties of the new algebra on the basis of the data semantics (Section 3.5).

[1] Consider, for example, the temporal Cartesian product. As in Codd's non-temporal algebra, it performs the concatenation of non-temporal attributes. As regards the temporal part, for each pair of input tuples, it has to return a unique time which is the combination of the times of the two tuples being concatenated. For semantic reasons widely discussed, e.g., in [1,5], the operation used for this combination is, in the case of Cartesian product, the intersection of times. Notably, the concatenations of the tuples whose temporal intersection is empty are not part of the output

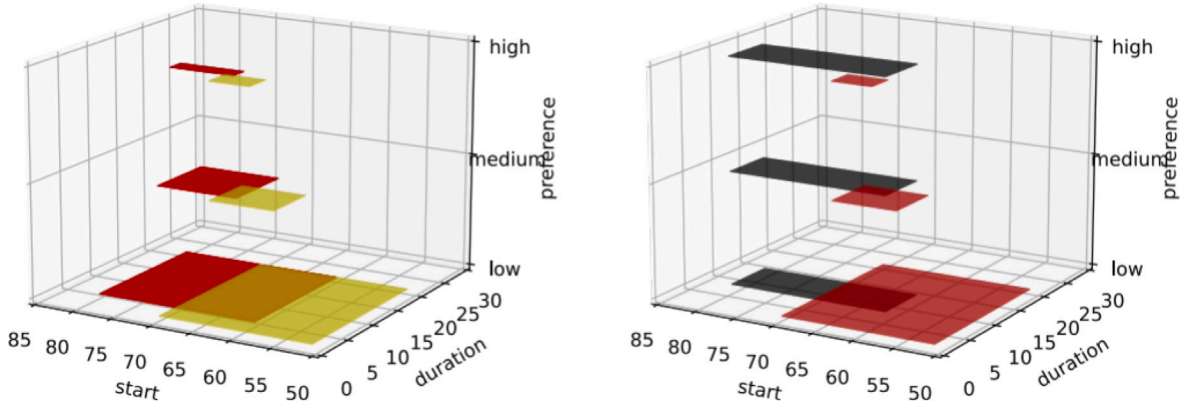


Fig. 2. Temporal Cartesian product (orange pyramid intersecting the red and the yellow pyramid on the left) and temporal difference (red pyramids excluding the black one on the right) [20]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.1. Motivations for an extension and problems to be addressed

The approach in [20], summarized in Section 2 above, allows one to associate layered temporal preferences with temporally indeterminate valid time, expressed as a (range of values for the) starting time and a (range of values for) duration—see, e.g., Ex. 2 above.

However, in complex cases, *starting time and duration* might not be enough to express temporal constraints on the validity of temporally indeterminate facts: also a range of possible values for ending time might be required, as in Ex. 2'.

To understand the relevance of considering also the ending time, we highlight the fact that ASBRU's temporal formalism [24], one of the reference approaches to time in literature on Artificial Intelligence in Medicine (see, e.g., the surveys in [25,26]), models times through a sextuple $\langle [ESS, LSS], [EFS, LFS], [minDu, maxDu] \rangle$, to represent that the action/fact must start in a time range $[ESS, LSS]$, end in a time range $[EFS, LFS]$, and the minimum duration of the action is $minDu$ and its maximum duration is $maxDu$. This is due to the fact that constraints considering starting time, ending time and duration are strictly more expressive than those considering only starting time and durations, or only starting time and ending time. Consider, for instance, Ex. 5.

Ex. 5. A fact f ***starts*** at 1, 2 or 3, ***ends*** at 4 or 5, and has ***duration*** 2 or 3. ■

Ex. 5 means that four different occurrences of f are possible (for the sake of brevity, we indicate each occurrence as a triple $\langle start, end, duration \rangle$): $\langle 1,4,3 \rangle$, $\langle 2,4,2 \rangle$, $\langle 2,5,3 \rangle$, and $\langle 3,5,2 \rangle$.

There is no way to capture such a temporal constraint considering starting time and duration only as we did in [20], or considering starting and ending time only as many other approaches in the literature. On the one hand, considering *start and duration* only, we can state that f starts at 1, 2 or 3, and lasts 2 or 3, but such a constraint allows, besides the four occurrences above, also the occurrences $\langle 1,3,2 \rangle$ and $\langle 3,6,3 \rangle$, which are excluded by Ex. 5. On the other hand, considering *start and end* only, we can state that f starts at 1, 2 or 3, and ends at 4 or 5, but such a constraint allows, besides the four occurrences above, also $\langle 1,5,4 \rangle$ and $\langle 3,4,1 \rangle$, which are excluded by Ex. 5.

However, if one wants to build a theoretically grounded framework, the addition of the ending time involves the analysis and the solution of quite subtle and intriguing problems. In fact, the end time is not an orthogonal dimension with respect to start time and duration (since duration is the distance between end and start time). Indeed, Ex. 5 above shows that the addition of (a range of values for) the end time reduces the number of possible occurrences (eliminating the pairs $\langle start, duration \rangle$ such that start+duration is not included in the range of possible ending times). Since, for the sake of efficiency, we still want to define a compact representation (not making explicit all the possible occurrences) and to define algebraic operators operating directly on it, we have to be careful in the definition of the semantics of the representation (which has to capture the fact that not all combinations of temporal values are possible) and of the operators and, above all, we have to prove the correctness of our new algebraic operators with respect to the semantics. Such challenging issues are addressed in the rest of this Section.

3.2. Data model

Concerning the temporal domain, and scales of preferences, we maintain the assumptions used in [20], and described at the beginning of Section 2. In particular, we denote by T^C the domain of chronons and by S_r a scale of preferences considering r different (and ordered) values for preferences. We also indicate by $S_r(i)$ (with $1 \leq i \leq r$) the ith value in the scale of preference S_r (e.g., if $S_3 = \langle low, medium, high \rangle$, $S_3(2) = medium$).

A temporal relation with layered preferences associates facts with valid times and preferences. For short, we term such relations HP (Hyperpyramid Preference)[2] relations.

Definition 3.1 (*HP Relation*)**.** Given a scale S_r, which is unique for a given DB, the schema of a HP relation $R = (\overline{A}|\overline{T}_r)$ consists of an arbitrary number of non-temporal attributes $\overline{A} = (A_1, \ldots, A_n)$, encoding some fact, and, separated by a symbol |, of r sextuples with temporal attributes $\overline{T}_r = ((T_{sm} : T_{sM}, T_{em} : T_{eM}, T_{dm} : T_{dM})^1, \ldots, (T_{sm} : T_{sM}, T_{em} : T_{eM}, T_{dm} : T_{dM})^r)$ with $T_{sm}, T_{sM}, T_{em}, T_{eM} \in T^C$, and $T_{dm}, T_{dM} \in \mathbb{N}$. We term each $(T_{sm} : T_{sM}, T_{em} : T_{eM}, T_{dm} : T_{dM})^i$ the ith Temporal Layer of the relation. ■

Thus, a tuple $x = (v_1, \ldots, v_n | (s^1_m : s^1_M, e^1_m : e^1_M, d^1_m : d^1_M)^1, \ldots, (s^r_m : s^r_M, e^r_m : e^r_M, d^r_m : d^r_M)^r)$ in a HP relation $r^{HP}(\overline{A}|\overline{T}_r)$ represents that the fact $(v_1, \ldots, v_n)$, with preference at least $S_r(1)$, starts between s^1_m and s^1_M, ends between e^1_m and e^1_M, and has duration between d^1_m and d^1_M, …, and, with preference at least $S_r(r)$, starts between s^r_m and s^r_M, ends between e^r_m and e^r_M, and has duration between d^r_m and d^r_M. Notably, since we adopt layered preferences [20], each range is included in the corresponding ranges at lower levels.

Ex. 6. Ex. 2' can be represented, using the scale $S_3 = \langle low, medium, high \rangle$, by the relation pat_treatm^{HP} with schema $(patient\_t, treatment|\overline{T}_3)$ as following[3]: $pat\_treatm^{HP} = \{(Mary, phys|(60 : 81,67 : 90,4 :$

[2] The reason of the name is that, while in the approach in Section 2 preferences form a sort of pyramid of nested rectangles (see, e.g., Fig. 1), here we add an additional dimension, so that, as we will see in the following, we will obtain a pyramid of nested cuboids, that we term hyperpyramid.

[3] For clarity and brevity, when unambiguous, in the superscript of a temporal layer we indicate the label $Sr(i)$ and not the index i (e.g., low and not 1).

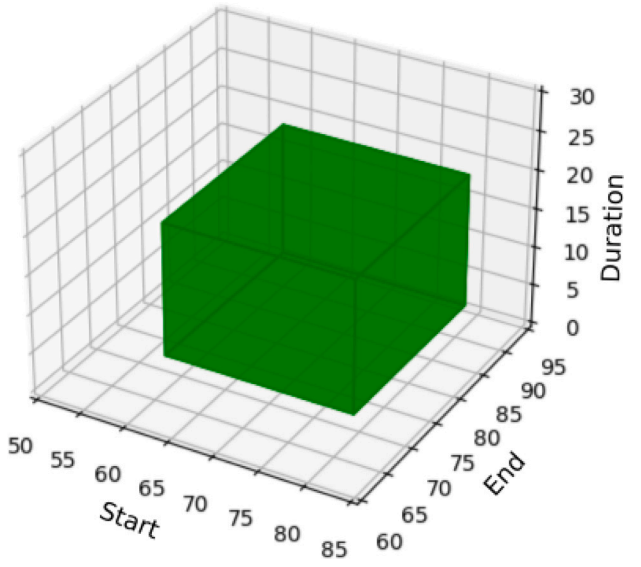


Fig. 3. Rectangular cuboid corresponding to preference level *low* for Ex. 2'.

$21)^{low}, (63:74, 68:87, 5:14)^{medium}, (64:73, 70:81, 6:9)^{high}$), stating the fact that Mary's physical therapy must, (i) with preference at least $S_r(1) = low$, start between 60 and 81, end between 67 and 90, last between 4 and 21, (ii) with preference at least $S_r(2) = medium$, start between 63 and 74, end between 68 and 87, last between 5 and 14, and (iii) with preference at least $S_r(3) = high$, start between 64 and 73, end between 70 and 81, last between 6 and 9. ■

The graphical representation of the temporal and preference component of tuples in a HP relation is less intuitive than the representation in the formalism in Section 2. The reason is that in Section 2 we worked on a three-dimensional representation, associating preferences with pairs $\langle start, duration\rangle$ (i.e., to rectangles). Here we have a four-dimensional representation, in which preferences are associated with triples $\langle start, end, duration\rangle$. In Fig. 3, we just show the parallelepiped (more precisely: a rectangular cuboid) determined by the coordinates $start \in [60, 81)$, $end \in [67, 90)$, $duration \in [4, 21)$, corresponding to the times with level of preference *low* in Ex. 2'.

3.3. Data semantics

HP relations support a compact and implicit representation of the association of valid times and preferences with facts, since a sextuple is used to summarize all the possible times of occurrence of a fact (for a given level of preference). *Following the methodology in [11]*, we formally define the semantics for HP relations through the definition of the functions *Ext* and *MakeExplicit*. The function *Ext*, given a Temporal Layer $(s^i_m : s^i_M, e^i_m : e^i_M, d^i_m : d^i_M)^i$ taken from the temporal part of a tuple in a HP relation, and corresponding to the ith level of the scale of preference, makes explicit all the pairs $\langle starting\_point, ending\_point, duration\rangle$ in the indeterminate valid time that have preference at least $S_r(i)$.

Definition 3.2 (*Data Semantics for a HP Relation: Ext* ***function***). Let $V_r = \langle (s^1_m : s^1_M, e^1_m : e^1_M, d^1_m : d^1_M)^1, \ldots, (s^r_m : s^r_M, e^r_m : e^r_M, d^r_m : d^r_M)^r\rangle$ be the temporal values associated with a tuple in a HP relation with time scale S_r, and let $V_r(i)$ denote the ith ($1 \le i \le r$) Temporal Layer (i.e., $(s^i_m : s^i_M, e^i_m : e^i_M, d^i_m : d^i_M)^i$), $Ext((s^i_m : s^i_M, e^i_m : e^i_M, d^i_m : d^i_M)^i) = \{\langle s, e, d\rangle \setminus s^i_m \le s < s^i_M \wedge e^i_m \le e < e^i_M \wedge d^i_m \le d < d^i_M \wedge s + d = e\}$. ■

Ex. 7. For example, $V_r(1) = (60:81, 67:90, 4:21)^{low}$ compactly represents the Temporal Layer $(s^1_m : s^1_M, e^1_m : e^1_M, d^1_m : d^1_M)^{low}$ having at least preference *low* for the tuple in *pat_treatm* (see Ex. 4).

The *Ext* function makes all such pairs explicit: $Ext((60:81, 67:90, 4:21)^{low}) = \{\langle 60, 67, 7\rangle, \langle 60, 68, 8\rangle, \ldots, \langle 80, 89, 9\rangle\}$. ■

The function *MakeExplicit*, applied to a HP relation r^{HP}, provides as output a new relation r^{Expl} in which each fact $x = (v_1, \ldots, v_n)$ in r^{TP} is paired with the explicit set of all the quadruples $\langle s, e, d, p\rangle$ such that x, starting at s, ending at e, and lasting d, has at least preference p.

Definition 3.3 (*Data Semantics for HP Relations:* ***MakeExplicit Function***). Given a HP relation r^{HP} over a schema $R = (\overline{A}|\overline{T}_r)$ and a scale S_r, *MakeExplicit* provides as output a new relation r^{Expl} defined over the schema $R' = (\overline{A}|\overline{S})$, where $\overline{S}$ is a set of quadruples $\langle s, e, d, p_i\rangle$ with $s, e \in T^C$, $d \in \mathbb{N}$, $p_i \in dom(S_r)$ defined as follows:

$r^{Expl} = MakeExplicit(r^{HP}) = \{(v_1, \ldots, v_n | \{\langle s, e, d, p_i\rangle \setminus \langle s, e, d\rangle \in Ext(V_r(i)), 1 \le i \le r \wedge p_i = S_r(i)\}) \setminus (v_1, \ldots, v_n | V_r) \in r^{HP}\}$.

For example, $MakeExplicit(pat\_treatm)$ makes explicit all the pieces of information denoted by *pat_treatm*, i.e., it associates with each fact all the quadruples denoting its starting time, ending time, duration, and preference.

3.4. Relational algebra

To define a query language for our data model, we provide a temporal relational algebra. In the definition of our operators, we follow the main principles emerging from the specialized literature, as discussed in Section 2, and that we used in the definition of the algebra for our *start + duration + preference* approach in [20].

However, there is a major difference between the approach in [20] and the one discussed here: we now add an additional dimension, the ending time. As a result, the new algebraic operators are applied to four-dimensional temporal data. In particular, the operation of Cartesian product and difference have now to perform intersection and difference between hyperpyramids, instead of pyramids.

In the following, we report the definition of the new algebraic operators, extending Codd's operators to work on our four-dimensional representation of preferential time. In Fig. 4 we report a notation guide.

Definition 3.4 (*Temporal Algebraic Operators for HP Relations*). Given a DB defined over the scale S_r, let r^{HP} and s^{HP} denote HP relations having the proper schema. In this definition v, $v1$, $v2$ stand for the values of non-temporal attributes in a tuple, and V_r, $V1_r$, $V2_r$ for the values of the temporal attributes—having the general form $\langle (s^1_m : s^1_M, e^1_m : e^1_M, d^1_m : d^1_M), \ldots, (s^r_m : s^r_M, e^r_m : e^r_M, d^r_m : d^r_M)\rangle$,

$\sigma^{HP}_P(r^{HP}) = \{(v|V_r) \setminus (v|V_r) \in r^{HP} \wedge P(v)\}$

$\pi^{HP}_A(r^{HP}) = \{(v'|V_r) \setminus \exists v, V_r((v|V_r) \in r^{HP} \wedge v' = \pi^{Codd}_A(\{v\}))\}$

$r1^{HP} \cup r2^{HP} = \{(v|V_r) \setminus (v|V_r) \in r1^{HP} \vee (v|V_r) \in r2^{HP}\}$

$r1^{HP} \times^{HP} r2^{HP} = \{(v1 \cdot v2|V_r) \setminus \exists V1_r, V2_r((v1|V1_r) \in r1^{HP} \wedge (v2|V2_r) \in r2^{HP} \wedge V_r = V1_r \cap^{hp} V2_r \wedge V_r \neq \emptyset)\}$

$r1^{HP} -^{HP} r2^{HP} = \{(v|V_r) \setminus \exists (v|V1_r) \in r1^{HP} \wedge V_r \in (\{V1_r\} -^{hp}_{n\_to\_m} \{V2_r \setminus (v|V2_r) \in r2^{HP}\}) \wedge V_r \neq \emptyset\}$

$\sigma^{HP}_{\text{start}=s,\text{end}=e,\text{duration}=d,\text{preference}=p}((v|V_r) \setminus (v|V_r) \in r^{HP} \wedge S_r(i) = p \wedge V_r(i) = (s^i_m : s^i_M, e^i_m : e^i_M, d^i_m : d^i_M)^i \wedge s^i_m \le s \le s^i_M \wedge e^i_m \le e \le e^i_M \wedge d^i_m \le d \le d^i_M \wedge s + d = e\}$

Cartesian product and difference operators use the $\cap^{hp}$ and $-^{hp}_{n\_to\_m}$ auxiliary operators for determining the intersection between hyperpyramids and difference between (sets of) hyperpyramids. They are defined in Appendix A.

In each operator, for each Temporal Layer $V_r(i)$, if $Ext(V_r(i)) = \emptyset$, then $V_r(i) = (0:0, 0:0, 0:0)^i$. Notice that in the formulas $V_r = \emptyset$ stands for $V_r = \langle (0:0, 0:0, 0:0)^1, \ldots, (0:0, 0:0, 0:0)^r\rangle$. ■

As motivated in Section 2, *selection* σ^{HP}_P, *union* $\cup^{HP}$, and *projection* π^{HP}_A only operate on non-temporal attributes, and behave like Codd's non-temporal algebraic operators (see, e.g., [1,5]), while $\times^{HP}$ and

In a HP tuple $x = (v_1, \ldots, v_n | (s^1_m : s^1_M, e^1_m : e^1_M, d^1_m : d^1_M)^1, \ldots, (s^r_m : s^r_M, e^r_m : e^r_M, d^r_m : d^r_M)^r)$:

- $v_1, \ldots, v_n$ are the explicit (non-temporal) values,
- the superscripts $1, \ldots, r$ indicate the preference level,
- s^i_m are the earliest starting times,
- s^i_M are the latest starting times,
- e^i_m are the earliest ending times,
- e^i_M are the latest ending times,
- d^i_m are the shortest durations,
- d^i_M are the longest durations,

Fig. 4. Notation reference.

$-^{HP}$ operate like Codd's operators on the non-temporal attributes and manipulate the temporal attributes.

Cartesian product $\times^{HP}$ performs the concatenation of the non-temporal parts of the tuples and performs the intersection of the two hyperpyramids modeling the temporal parts of the tuples using the $\cap^{hp}$ operation. Notably, the intersection of two hyperpyramids can be empty or a (possibly degenerate) hyperpyramid.

Relational difference $r1^{HP} -^{HP} r2^{HP}$ gives as output all the tuples $(v|V1_r) \in r1^{HP}$ that have no *value-equivalent* tuples (i.e., tuples with the same value v for the non-temporal attributes [1]) in $r2^{HP}$. Additionally, if $r2^{HP}$ contains tuples value equivalent to v, their temporal extent must be removed from the hyperpyramid $V1_r$.

Temporal selection $\sigma^{HP}_{\text{start}=s,\text{end}=e,\text{duration}=d,\text{preference}=p}$ operates on the temporal attributes obtaining the tuples that include the provided starting, ending times and duration at the given preference level.

3.5. Properties of the algebra

The temporal algebraic operators we defined in Section 3.4 manipulate an implicit representation. Thus, as motivated in Section 2, one has to prove their correctness. Intuitively, we have to prove that, for each algebraic operator Op^{HP} in our algebra, applying Op^{HP} to relations in our representation and applying the $MakeExplicit$ function to the result gives the same output obtained by moving from our representation to an explicit one, and then applying the "explicit" operator Op^{Expl} corresponding to Op^{HP}. Such a proof can be graphically schematized as shown in Fig. 5.

Obviously, for the sake of the proof, the algebraic operators working on the explicit representation (i.e., on the data semantics, see Section 3.3) must be defined. As an example, we define $\times^{Expl}$; the other operators are similar. $\times^{Expl}$ operates in the standard way on non-temporal attributes, and performs the intersection of the temporal parts of the tuples, represented by sets of quadruples $\langle s, e, d, p \rangle$.

Definition 3.5 (*Cartesian Product on the Explicit Representation*). $r1^{Expl} \times^{Expl} r2^{Expl} = \{(v1 \cdot v2|S) \backslash \exists S1, S2((v1|S1) \in r1^{Expl} \wedge (v2|S2) \in r2^{Expl} \wedge S = S1 \cap S2 \wedge S \neq \emptyset)\}$. ■

Property 3.1 (*Correctness of Time Interval Manipulation*). *The temporal algebraic operators Op^{HP} extending Codd's operators are correct: for each Op^{HP}, the following property holds where Op^{Expl} is the algebraic operator on the explicit model corresponding to Op^{HP}:*

$MakeExplicit(r1^{HP} Op^{HP} r2^{HP}) = MakeExplicit(r1^{HP})\ Op^{Expl}\ MakeExplicit(r2^{HP})$. ■

The proof is in Appendix B. The proof is similar to the one we proposed in [20], considering a temporal relational approach modeling starting time, duration and preferences. Here, however, we have to consider an additional dimension, the ending time, which is not orthogonal to the other two (see the discussion in Section 3.1).

The property of ***reducibility*** is a fundamental one for all TDB approaches, since it grants that the new temporal operators, which extend simpler operators to cope with new phenomena, are a consistent extension of simpler operators, in the sense that they reduce to simpler operators when the new phenomena are disregarded [1,5].

To prove reducibility, standard TDB approaches introduce a time-slice operator that selects all the tuples holding at a specific time, and then remove the temporal attribute(s) from them [1,5]. In our extension, it considers four parameters: a starting time s, an ending time e, a duration d, and a preference level i. Our time-slice operator $\tau^i_{\langle s,e,d \rangle}$ is defined as follows.

Definition 3.6 (*Time-Slice Operator* $\tau^i_{\langle s,e,d \rangle}$). Let r^{HP} a HP relation, defined on the schema $R = (\overline{A}|\overline{T_r})$, S_r a scale, $s, e \in T^C$ and $d \in \mathbb{N}$, and i $(1 \leq i \leq r)$ a preference level in S_r:

$\tau^i_{\langle s,e,d \rangle}(r^{HP}) = \{z \backslash \exists x \in r^{HP}(z[\overline{A}] = x[\overline{A}] \wedge \langle s, e, d \rangle \in Ext(V_r(i)))\}$. ■

Given the above definition, Property 3.2 holds.

Property 3.2 (*Reducibility to Codd's Algebra*). *The HP relational algebra is reducible to Codd's relational algebra, i.e., for each pair of HP relations $r1^{HP}$ and $r2^{HP}$ defined over a proper schema, and considering a scale S_r, for each $s, e \in T^C, d \in \mathbb{N}$, and preference level i $(1 \leq i \leq r)$ in the scale S_r the following holds:*

$\tau^i_{\langle s,e,d \rangle}(r1^{HP}\ Op^{HP}\ r2^{HP}) = \tau^i_{\langle s,e,d \rangle}(r1^{HP})\ Op^{Codd}\ \tau^i_{\langle s,e,d \rangle}(r2^{HP})$ *where Op^{HP} and Op^{Codd} represent corresponding relational operators in HP algebra and in Codd's algebra respectively.* ■

The proof is in Appendix B.

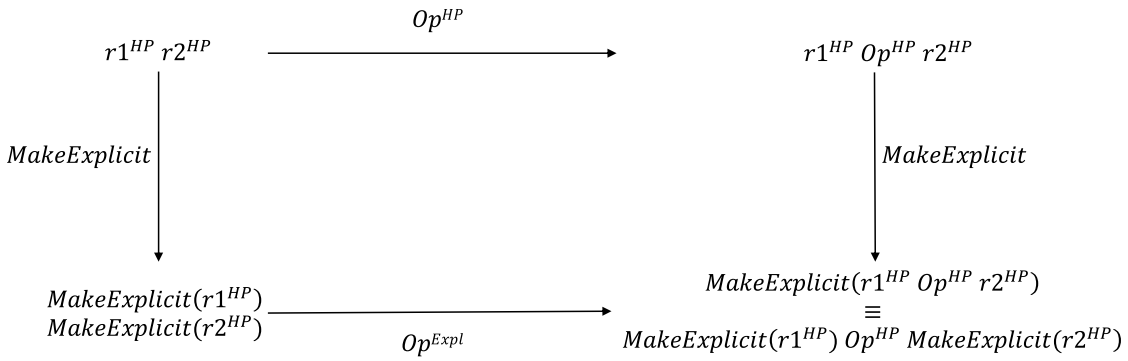


Fig. 5. Graphical representation of the correctness property of HP operators.

4. Experimental evaluation

In this Section, we analyze the performance of our approach considering different viewpoints, and comparing it with several other approaches. In particular, we consider seven approaches: *Hyperpyramid*, *Pyramid*, *Indet*, *Exact*, *No-Time*, *Explicit_Hyperpyramid*, and *Explicit_Pyramid*.

- *Hyperpyramid*: Implements the approach detailed in Section 3.
- *Pyramid*: Implements the approach described in Section 2.
- *Indet*: Handles temporal indeterminacy without preferences.
- *Exact*: Deals with precise time (no temporal indeterminacy, no preferences).
- *No-Time*: Handles only non-temporal attributes.
- *Explicit_Hyperpyramid, Explicit_Pyramid*: Constitute a "baseline": they propose an explicit representation of all the possible times and preferences which are compactly covered by the *Pyramid* and *Hyperpyramid* approaches.

In Section 4.1 we sketch our implementation of the seven approaches above, and discuss how we generated the datasets for the different tests. In Section 4.2 we focus only on the approaches considering preferences in a compact way, i.e. *Pyramid* (see Section 2) and *Hyperpyramid* (see Section 3). We compare such approaches analyzing the performance (execution time and I/O) of their temporal operators (Cartesian product, difference and temporal selection) when varying the dataset size and the number of preference levels. In Section 4.3, we consider again *Pyramid* and *Hyperpyramid* and compare their performance with respect to the *Explicit* approaches, to highlight the advantages of coping in an compact and implicit way with temporal indeterminacy and preferences. Finally, in Section 4.4 we compare *Pyramid* and *Hyperpyramid* with less expressive approaches (i.e., *Indet*, *Exact*, *No-Time*), to ascertain the overhead added to deal with preferences and indeterminate time.

4.1. Implementation and dataset

Our implementation[4] relied on PostgreSQL 16 and PL/pgSQL stored procedures, relying on native PostgreSQL features rather than external temporal extensions. Experiments were conducted on a machine with an Intel Core i7-10750H CPU (2.60 GHz), 32 GB RAM, running Ubuntu 24.04, and PostgreSQL 16 DBMS with default settings (effective cache size: 500 MB, page size: 8 KB, shared buffers: 500 MB).

[4] The experimental code and data is available at the GitHub repository https://github.com/TemporalDatabases-UNITO/Implementation-Evalutation-TRwLP.

To manage temporal data, we employed PostgreSQL *int4range* data type for representing temporal intervals, custom arrays and user-defined types for managing multiple temporal intervals simultaneously, and PL/pgSQL scripts for procedural data transformations and operators.

I/O block measurements were obtained using the PostgreSQL extension *pg_stat_kcache*, which monitors physical disk access. Each experiment was repeated five times, and average results were reported. To ensure reliable and reproducible results the tables were reinitialized prior to every experimental run.

4.1.1. Approaches considered in the evaluations

The seven implemented model variants are sketched below:

- *Exact*: The Exact variant mirrors the valid-time semantics of the TSQL2 temporal model. Every tuple uses *int4range* to represent an interval, thus aligning precisely with the single valid-time interval notion inherent in TSQL2. In our implementation we did not use the legacy TSQL2-specific extension that was instead simulated by using PostgreSQL's built-in interval operations, using the closed-open interval convention, and implementing the various operations entirely through native PostgreSQL queries.
- *Indet*: For simplicity, it is implemented as a single-preference level instantiation of the Pyramid approach.
- *Pyramid* and *Hyperpyramid*: These models represent hierarchical preferences with two *int4range*s (three in the case of hyperpyramid) for level of preference.
- *Explicit_Pyramid* and *Explicit_Hyperpyramid*: Tuples contain multiple intervals within an array; each assigned a preference level consistent with the Pyramid/Hyperpyramid models.
- *No-time*: Each tuple has only non-temporal attributes to establish a baseline for comparisons.

The operations were implemented across all variants through SQL combined with PL/pgSQL stored procedures.

4.1.2. Dataset generation

Datasets of varying sizes were generated to evaluate the scalability of our approach, focusing on the algebraic operators that manipulate temporal and preference attributes: Cartesian product and difference. To ensure reproducibility and precise control over experimental parameters, we designed a custom data generation procedure implemented in PL/pgSQL. It accepts several configurable parameters:

- *num_rows*: Total number of tuples generated.
- *nontemp_intersect*: Percentage of tuples sharing non-temporal attributes (Attr1, Attr2).
- *temp_intersect*: Percentage of tuples with overlapping temporal intervals.
- *interval_mul*: Factor to scale interval size.
- *seed*: Random seed for data generation.

Explicit variants utilize a post-processing function (*MakeExplicit*) to create interval arrays with associated preference levels representing the explicit version of temporal attributes of the Pyramid or Hyperpyramid tuples.

4.1.3. Table definition

We used tables with two non-temporal varying-char attributes (25 characters each). This setup represents an unfavorable scenario for our approach, as the temporal attributes added per tuple are proportional to the number of preference levels—four times for the Pyramid approach and six times for the Hyperpyramid approach. The tables we use to evaluate the algebraic operators in the *Hyperpyramid* and *Pyramid* approaches are:

- For evaluating the *Cartesian product* $T1 \times^{TP} T2$ ($T1 \times^{HP} T2$) two tables $T1$ and $T2$ were generated with increasing sizes (500, 1 000, 2 500, 5 000 tuples)[5]; preference levels ranged from 3 to 10 (3, 5, 10 preference levels); pyramid (and hyperpyramid) bases were of varying sizes, assuming $s_M - s_m = d_M - d_m$ ($s_M - s_m = e_M - e_m = d_M - d_m$) for simplicity (we call such difference *interval size*). The interval size was configured such that 10% of the tuples in $T1$ intersected with their value-equivalent counterparts in $T2$.
- For evaluating the *difference* $T1 -^{TP} T2$ ($T1 -^{HP} T2$), tables $T1$ and $T2$ were generated with increasing sizes (10 000 to 500 000 tuples); preference levels ranged from 3 to 10 (3, 5, 10 preference levels); 10% of the tuples in $T1$ had value-equivalent counterparts in $T2$, with 20% of these temporally intersecting.
- For evaluating the *temporal selection* $\sigma_P^{TP}(T)$ ($\sigma_P^{HP}(T)$), table T was generated with increasing sizes (10 000 to 500 000 tuples); preference levels ranged from 3 to 10 (3, 5, 10 preference levels); the selection condition was defined with a selectivity of approximately 20% (i.e., about 20% of the tuples in T satisfied the condition).

In the following, we detail the tables we used to test the other approaches:

- *Explicit*: In the explicit approach, the corresponding tables, $T1'$ and $T2'$, retain identical non-temporal attributes to $T1$ and $T2$ but explicitly represent temporal attributes as triples $\langle s, d, p\rangle$ for the Pyramid approach or quadruples $\langle s, e, d, p\rangle$ for the Hyperpyramid approach.
- *Indet*: We use the tables above, considering only one preference level. Thus, each tuple has two *int4ranges* to represent valid-time indeterminacy (start+duration) for identical non-temporal attributes.
- *Exact*: This approach uses the algebraic operators of TSQL2, with tables $T1''$ and $T2''$ (corresponding to $T1$ and $T2$), containing two temporal attributes (start and end of valid time). In this setup, 10% of the tuples temporally intersect.
- *No-Time*: Removes both time and preferences, using non-temporal Codd's operators [3]. The tables contain only the non-temporal attributes of tables $T1$ and $T2$.

4.2. Pyramid and hyperpyramid temporal operators

We evaluated the impact of preference levels on Cartesian product, difference and temporal selection using the described datasets with *interval size* 10.

Figs. 6, 7 and 8 present the execution times and I/O operations for varying dataset sizes and preference levels. These figures illustrate how performance metrics are affected by the use of 3, 5, and 10 preference levels. Our analysis explores the relationship between preference levels and system efficiency, providing insights into scalability and the resource requirements associated with varying preference levels.

For all three operations, increasing preference levels resulted in higher execution times and I/O blocks as more temporal information had to be processed because of the larger number of quadruples ($\langle min\_start : max\_start, min\_duration : max\_duration\rangle$) or sextuples ($\langle min\_start : max\_start, min\_end : max\_end, min\_duration : max\_duration\rangle$) required to represent temporal attributes.

Interestingly, while higher preference levels introduced additional complexity, the increase in execution time and I/O was not directly proportional to the number of levels. For instance, processing 10 preference levels did not lead to a tenfold increase in I/O compared to a single level. This is due to the fact that the temporal values are just a part of the overall data (which also consist of the values for non-temporal attributes) and the computation related to the temporal component (e.g., the computation of intersection for temporal Cartesian product) is just a part of the overall computations (e.g., concatenation of non-temporal values for temporal Cartesian product).

It is worth noting that, in practical applications, 5 preference levels might suffice, making the observed performance overhead acceptable for real-world scenarios.

Not surprisingly, the Hyperpyramid approach shows a higher cost with respect to the Pyramid one. This is mainly due to the fact that an additional pair of attributes $\langle min\ end : max\ end\rangle$ is added at each preference level, and additional operations to manage it are performed. In the case of Cartesian product, this fact reflects on a moderate increase of both execution time (three min/max operations have to be performed at each level, instead of two), and I/O blocks (two additional attributes at each level). In the case of difference, the addition of the ending time determines a growth (from four to six) of the possible pyramids that can be generated. Thus, at each level, 18 min/max operations have to be considered (three temporal dimensions times six hyperpyramids) versus 8 (two temporal dimensions times four pyramids) in the pyramid approach. This fact involves a significant increase of execution time, and a moderate increase of I/O blocks. In the case of temporal selection, the higher cost of the Hyperpyramid approach is due to the additional comparison per level and to extra filtering conditions involving both the end time and the duration (i.e., the difference between start and end). This results in a moderate increase in both execution time and I/O blocks.

4.3. Comparison with the explicit approach

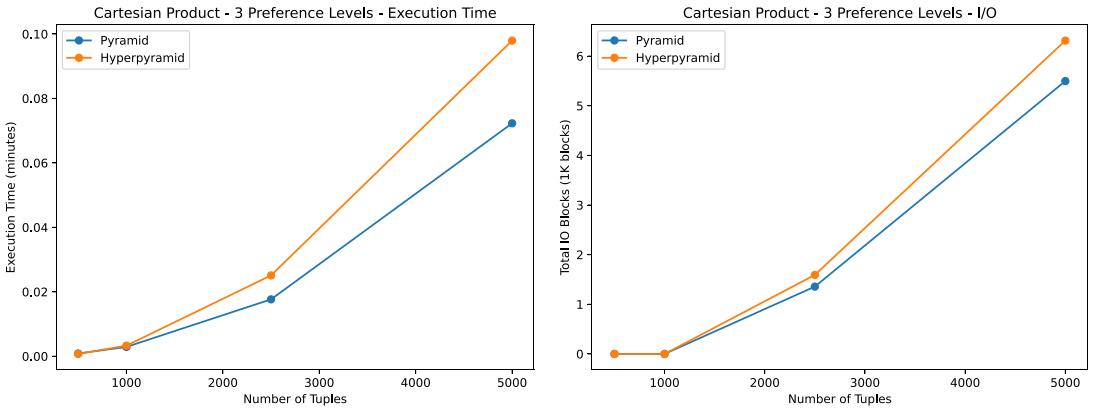
This section compares our implicit methods for handling temporal indeterminate intervals (Pyramid and Hyperpyramid) with their explicit counterparts. In the explicit approach, temporal indeterminate intervals are explicitly represented, as discussed in Sections 2.3 and 3.3 regarding data semantics and the $MakeExplicit$ function. In the implicit approach, the tables $T1$ and $T2$ are structured as previously described, with an *interval size* of 10.

This comparison aims to showcase differences in data representation and efficiency between implicit and explicit approaches, emphasizing their respective impacts on processing performance and data manageability.

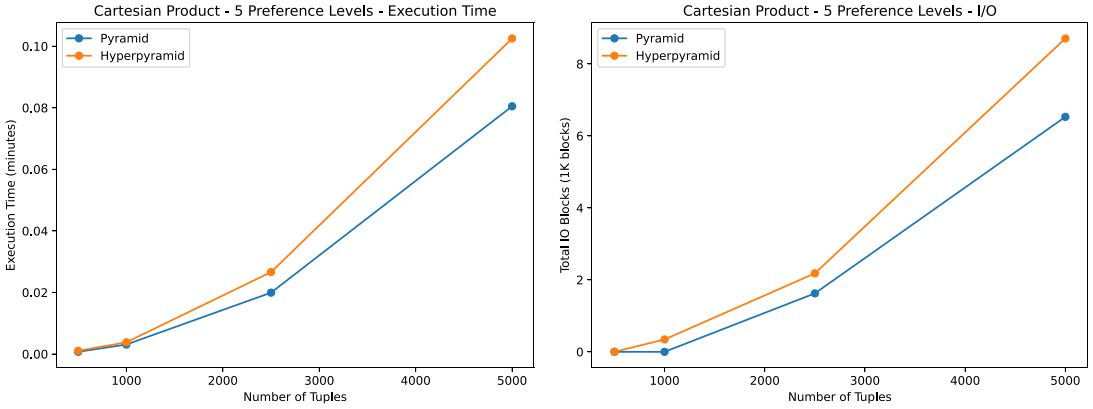
Figs. 9, 10 and 11 compare the performance of implicit and explicit approaches for varying preference levels in the context of the Cartesian product, difference and temporal selection operations, focusing on execution time and I/O performance.

The explicit approach consistently demonstrates higher execution times and I/O block usage compared to the Pyramid and Hyperpyramid approaches across all preference levels. This disparity is especially pronounced in the Cartesian product operation, where execution time and I/O usage for the explicit approach increase significantly with larger datasets.

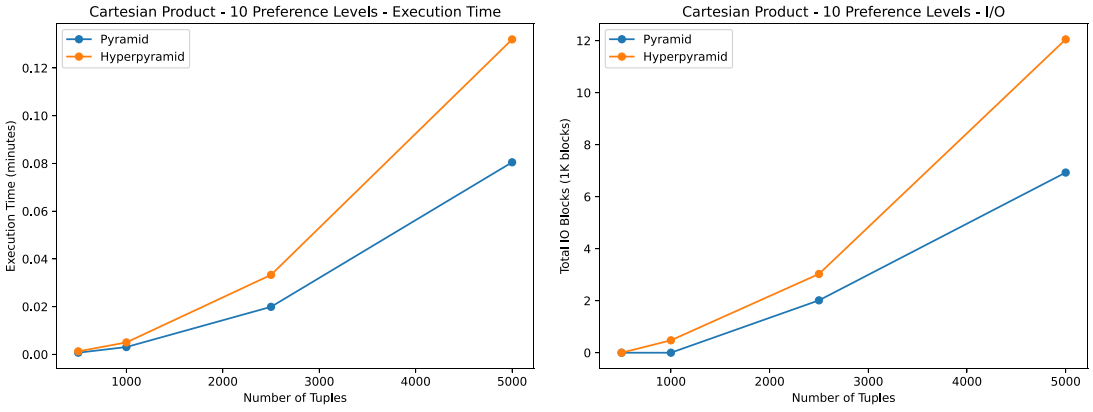
[5] Temporal Cartesian product performs pairwise concatenation of the non-temporal part of the tuples, and is thus quadratic (even though only those tuple concatenations such that the intersection of the time of the tuples is not empty are retained in the output). This the reason why we had to limit the number of tuples in the experiments for this operator.



(a) Three pref. levels - Exec. Time (b) Three pref. levels - I/O



(c) Five pref. levels - Exec. Time (d) Five pref. levels - I/O



(e) Ten pref. levels - Exec. Time (f) Ten pref. levels - I/O

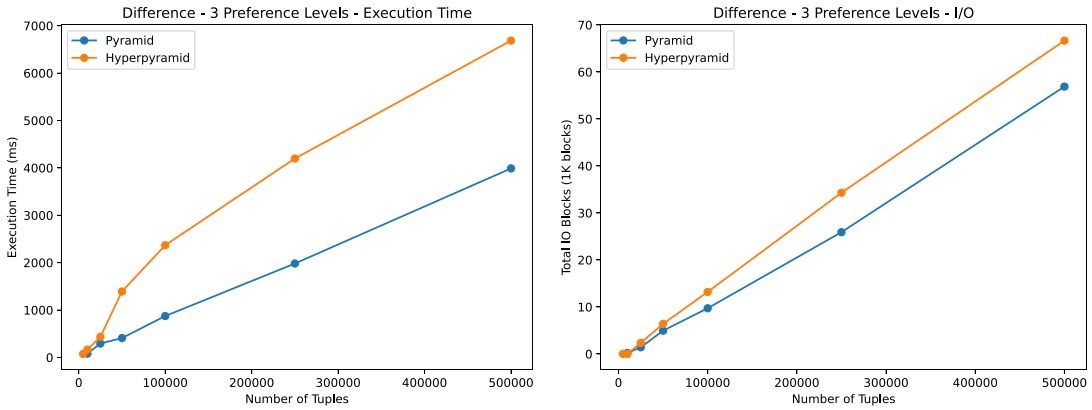
Fig. 6. Pyramids and Hyperpyramids (Cartesian product): comparisons in terms of execution time and I/O between Pyramid and Hyperpyramid approaches considering an increasing number of preference levels, with datasets of different sizes.

In contrast, the implicit approach shows a more gradual rise in execution time and I/O operations as the number of preference levels increases. This behavior underscores its better scalability and efficiency, particularly in handling large datasets. These results highlight the benefits of the implicit approach, especially in scenarios with high data volumes, where its ability to manage preferences more efficiently offers substantial performance advantages.

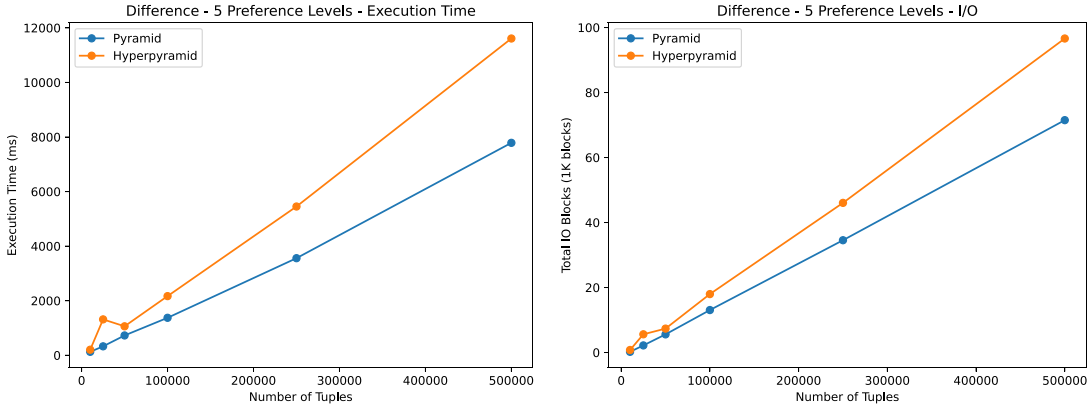
Notably, both the explicit and the implicit approaches are more expensive in the case of hyperpyramids with respect to pyramids, due to the additional temporal component and its related operations. However, also the gain when moving from an explicit approach to an implicit one increases with the complexity of the representation (i.e., it is greater for hyperpyramids).

Obviously, the advantages of our implicit approaches (with respect to explicit ones) grow together with the growth of the sizes of the intervals specifying possible indeterminacies.

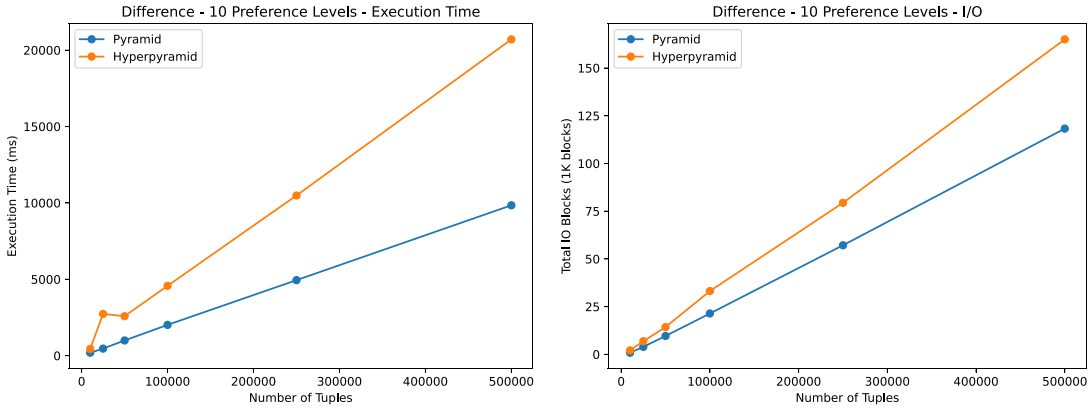
Figs. 12, 13 and 14 illustrate the effects of varying interval sizes on execution time and the number of I/O operations for implicit and explicit approaches, focusing on data with three preference levels.



(a) Three pref. levels - Exec. Time (b) Three pref. levels - I/O



(c) Five pref. levels - Exec. Time (d) Five pref. levels - I/O



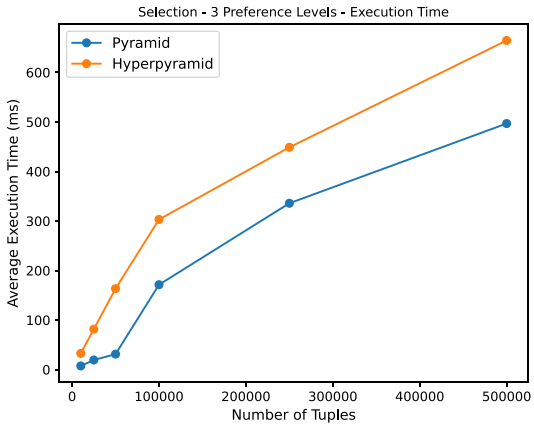
(e) Ten pref. levels - Exec. Time (f) Ten pref. levels - I/O

Fig. 7. Pyramids and Hyperpyramids (difference): comparisons in terms of execution time and I/O between Pyramid and Hyperpyramid approaches considering an increasing number of preference levels, with datasets of different sizes.

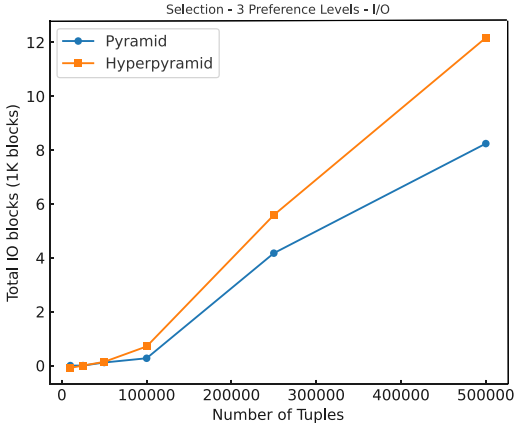
These results are critical for understanding how performance is influenced by interval size, particularly in the context of Cartesian product operations. The interval sizes examined are 10, 50, and 100. To ensure the feasibility of the experiments, a maximum runtime of one hour was imposed, leading to censored results for the explicit approach in cases where the interval size was 50 for datasets with 5 000 tuples and 100 for datasets exceeding 1 000 tuples. These limitations are reflected in Fig. 12 and underscore the explicit approach's challenges in handling larger intervals.

The results indicate that, in the Pyramid and Hyperpyramid implicit approaches, both execution time and I/O operations remain unaffected by interval size. Thus, only the solid green line (interval size 100) is visible in Figs. 12 and 13. The stability in performance demonstrates the implicit approaches' ability to handle larger datasets without significant resource strain.

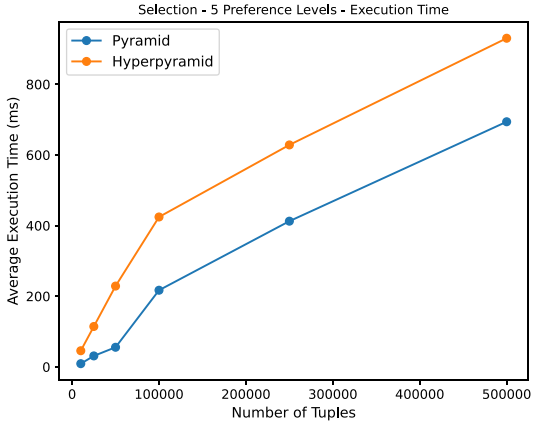
In contrast, there is a notable performance degradation for explicit approaches as execution time and I/O operations increase significantly with larger interval sizes. This is due to the explicit representation of intervals, requiring more data for triples $\langle start, duration, preference \rangle$



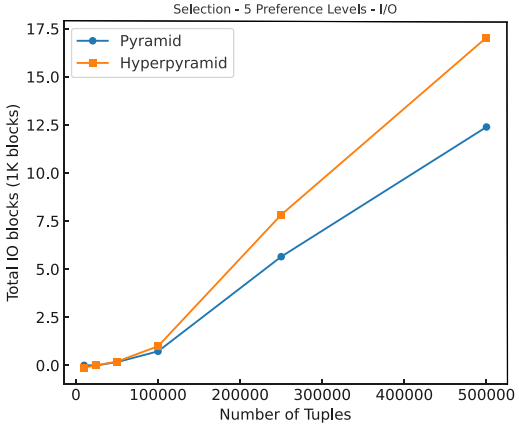
(a) Three pref. levels - Exec. Time



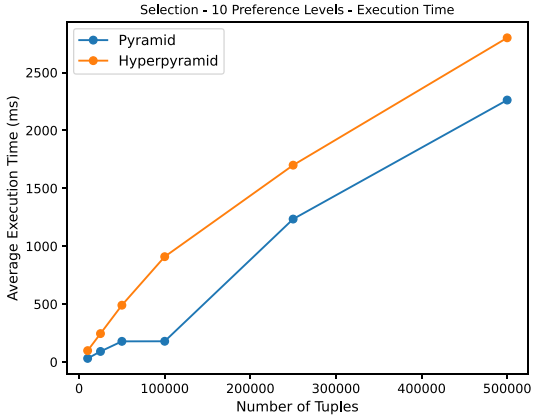
(b) Three pref. levels - I/O



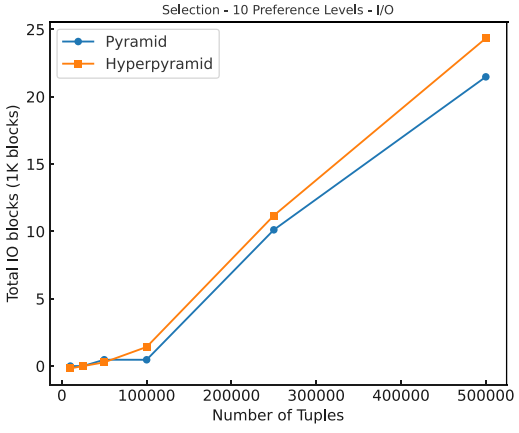
(c) Five pref. levels - Exec. Time



(d) Five pref. levels - I/O



(e) Ten pref. levels - Exec. Time



(f) Ten pref. levels - I/O

Fig. 8. Pyramids and Hyperpyramids (temporal selection): comparisons in terms of execution time and I/O between Pyramid and Hyperpyramid approaches considering an increasing number of preference levels, with datasets of different sizes.

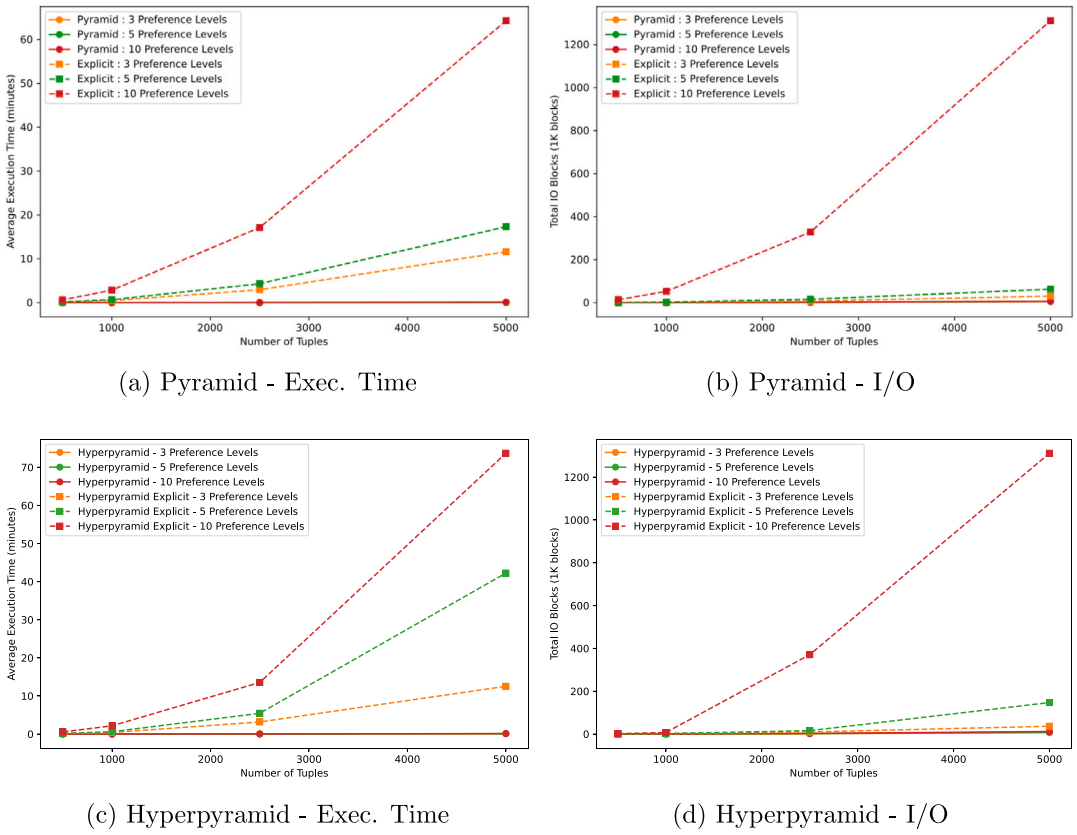


Fig. 9. Comparison with the Explicit Approach (Cartesian product): comparisons between implicit and explicit approaches for Pyramid and Hyperpyramid considering execution time and I/O across varying preference levels and dataset sizes.

(or quadruples $\langle start, end, duration, preference \rangle$ in the Hyperpyramid approach). The pronounced overhead highlights potential scalability issues, especially for large datasets or when managing extensive intervals.

The comparison reveals a stark contrast between implicit and explicit approaches. Implicit approaches demonstrate better adaptability and minimal impact on system resources, even with larger interval sizes. In contrast, explicit methods impose a substantial burden as interval sizes grow, limiting their efficiency and scalability in high-demand scenarios. This highlights the implicit approach as a more robust solution for managing temporal data with extensive intervals.

4.4. Overhead with respect to less expressive approaches

We now evaluate our approach by comparing it to less expressive methods, highlighting the trade-offs introduced by the added expressiveness of managing *indeterminate time* and *preferences*. This comparison explores the performance overhead of our approach relative to methods that lack one or both features.

We report the results of experiments for Cartesian product in Fig. 15. On the one hand, adding exact or indeterminate time introduces negligible overhead compared to the non-temporal approach. On the other hand, both Pyramid and Hyperpyramid approaches exhibit moderate overhead, primarily due to the increased tuple size (i.e., 12 additional attributes for Pyramid and 18 for Hyperpyramid with three preference levels) and additional interval intersection computations. The main cost driver remains the Cartesian product's inherent complexity—combining all tuple pairs in the input tables.

The experiments on the difference and temporal selection operations (as illustrated in Figs. 16 and 17) largely corroborate the findings observed for the Cartesian product. However, in the case of difference, the non-temporal approach significantly outperforms temporal methods in execution time and I/O. This is due to the absence of temporal operations and smaller tuple sizes (i.e., no temporal attributes compared to 12 or 18 temporal attributes in Pyramid and Hyperpyramid approaches; notably, in the experiments we consider just two non-temporal attributes). Moreover, temporal difference produces larger outputs as value-equivalent tuples can also contribute, further increasing I/O overhead. Exact and Indet approaches show similar performance, with additional overhead observed for Pyramid and Hyperpyramid due to the larger number of temporal attributes and the corresponding increase in operations. The case of temporal selection is similar. Notably, however, we could not consider the non-temporal approach as a baseline, since in such a context temporal selection is meaningless.

5. Related work

As already mentioned in the introduction, while temporal preferences are important in several domains, including medicine and economy, no other approach in the temporal database literature provides the possibility to cope with temporal preferences. As a consequence, since temporal preferences apply to scenarios in which time is indeterminate, the approaches in the temporal database literature closest to ours are those coping with temporal indeterminacy.

Two landmark approaches in the literature deal with temporal indeterminacy in combination with probabilities: Dekhtyar et al. [7] and by Dyreson and Snodgrass [6]. Dekhtyar et al. introduce temporal probabilistic tuples to model the probability of tuples to exist in a given relation r at some point of time within a time period, and define temporal algebraic relational operators to query their data model.

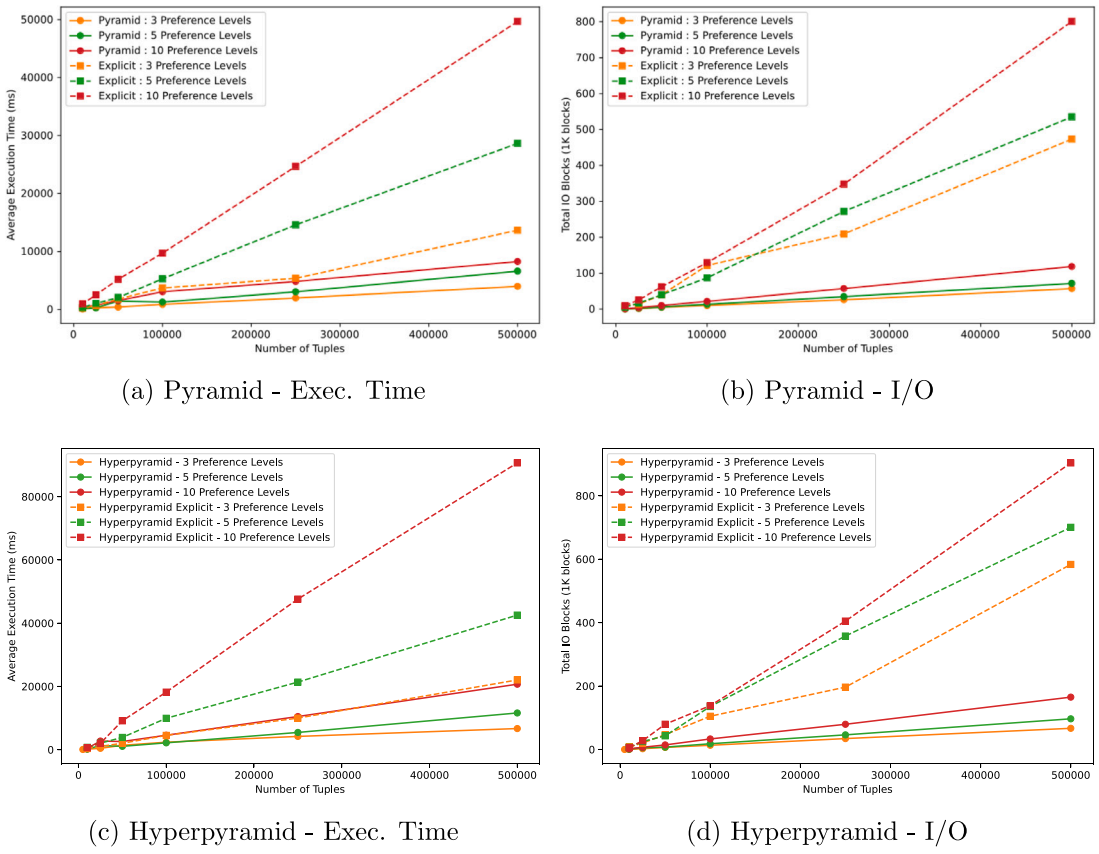


Fig. 10. Comparison with the Explicit Approach (difference): comparisons between implicit and explicit approaches for Pyramid and Hyperpyramid considering execution time and I/O across varying preference levels and dataset sizes.

Notably, however, Dekhtyar et al. restrict their attention to events that are instantaneous, while our approach also considers durative events (of course, we can deal with instantaneous events as events starting and ending at the same time).

On the other hand, Dyreson and Snodgrass [6] deal with temporal indeterminacy by associating a period of indeterminacy with a tuple. A period of indeterminacy is a period between two indeterminate instants, each one consisting of a range of chronons (modeling the basic granularity of the temporal database) and of a probability distribution over it. While, differently from [7], Dyreson and Snodgrass' data model copes with durative events, their approach does not provide temporal algebraic operators to query it.

In [8], some of the authors of this paper, in cooperation with R.T. Snodgrass, propose a reference model and algebra for an explicit treatment of valid-time indeterminacy (e.g., making explicit all the possible occurrences of events), and then specify, analyze and compare a family of sixteen more compact implicit representations. Additionally, [8] extends the reference model and the compact representation models to cope with probabilities.

A different stream of approaches, in the line of Artificial Intelligence research, model temporal indeterminacy in the context of temporal constraints between tuples, with specific attention to relative times (consider, e.g., Brusoni et al. [27] and Koubarakis [28]).

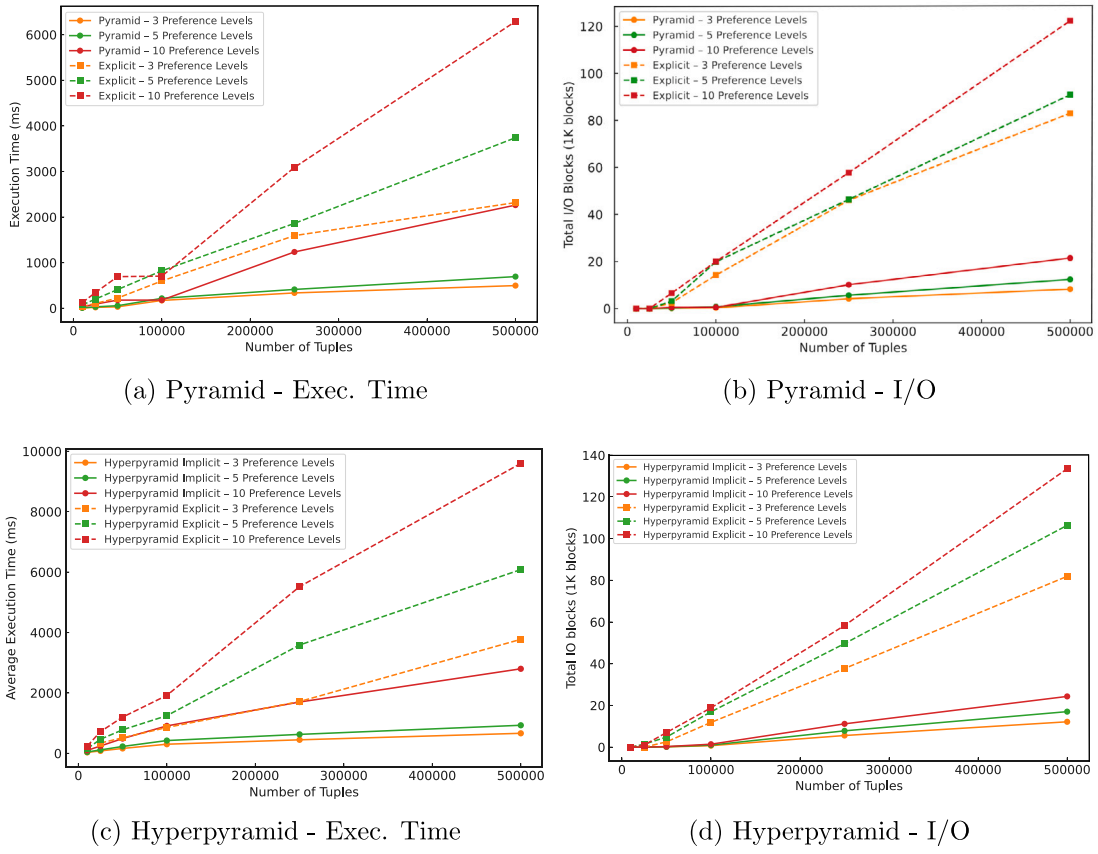
As already discussed in the introduction, our work on incorporating preferences into temporal relational databases has started with [20], where we defined a data model and algebra to deal with start time and duration with preferences, and proved the correctness and reducibility of our algebra. In our ADBIS paper [29] we then proposed a performance evaluation of such a theoretical approach. While this paper is grounded on such a background, it extends our previous work to consider also ending times. Such an extension involved the solution of challenging problems (see the informal discussion in Section 3.1) and proposes four main original contributions:

(1) A new data model (to consider also ending time) and the specification of its semantics (see Sections 3.2 and 3.3),
(2) A new temporal algebra (to consider also ending time), and the proofs of its correctness and reducibility (see Section 3.4, Appendices A and B),
(3) The experimental evaluation of the new approach, and its comparison with less expressive approaches (including [20]; see Section 4),
(4) The definition of the operator of temporal selection, its performance evaluation, and its comparison with temporal selection in less expressive approaches.

6. Conclusions

In many tasks and applications, the exact timing of events is often unknown, leading to *temporal indeterminacy*, and *preferences* can be assigned to various temporal possibilities. In a recent paper, we established the theoretical foundation for the first TDB approach that supports preferential indeterminate time, focusing on the starting time and duration of events.

In this paper, we extended this work by introducing an additional temporal dimension, the *ending time*, which enhances expressiveness as illustrated in Ex. 2'. We proposed a new data model and algebra to accommodate this feature, discussed the semantics of the model, and proved the correctness and reducibility of the new algebra.



(a) Pyramid - Exec. Time (b) Pyramid - I/O

(c) Hyperpyramid - Exec. Time (d) Hyperpyramid - I/O

Fig. 11. Comparison with the Explicit Approach (temporal selection): comparisons between implicit and explicit approaches for Pyramid and Hyperpyramid considering execution time and I/O across varying preference levels and dataset sizes.

Furthermore, we conducted a performance evaluation of both approaches across various dimensions and analyzed the overhead introduced by managing preferences compared to relational approaches that (i) lack temporal data, (ii) include exact temporal data, and (iii) incorporate indeterminate time without preferences. The experimental results demonstrate that, while the introduction of multiple preference levels increases expressiveness, it imposes minimal additional overhead. This indicates that incorporating indeterminate time management capabilities significantly enhances the representational capacity of the data with only a modest impact on performance. Notably, the added expressiveness of including ending points in the model has a negligible effect on the overall performance of the framework.

For future work, we plan to take the necessary steps to propose a full extension of PostgreSQL to manage temporal preferences, leveraging the implementation of temporal algebraic operators discussed in this paper. Our first objective will be to extend SQL to support the creation and querying of tables with temporal preferences. Following the methodology outlined in Section 24 of the TSQL2 book [1], subsequent steps will include enhancing the data dictionary, DDL, query compiler, runtime evaluator, and transaction and data manager. These enhancements may build upon existing extensions like those provided by https://github.com/scalegenius/pg_bitemporal, which manage (bi)temporal tables in PostgreSQL.

Additional avenues for future research include extending the framework to handle real-time data streams, integrating machine learning techniques to discover and adapt users preferences based on their behavior, developing a comprehensive benchmarking suite for testing TDB approaches, and exploring the incorporation of other forms of temporal indeterminacy, potentially modeled using probability distributions.

CRediT authorship contribution statement

Luca Anselma: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Antonella Coviello:** Writing – review & editing, Writing – original draft, Software, Data curation. **Davide Cerotti:** Writing – review & editing, Validation, Software, Investigation, Conceptualization. **Erica Raina:** Writing – review & editing, Writing – original draft, Visualization, Software, Formal analysis, Conceptualization. **Paolo Terenziani:** Writing – review & editing, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

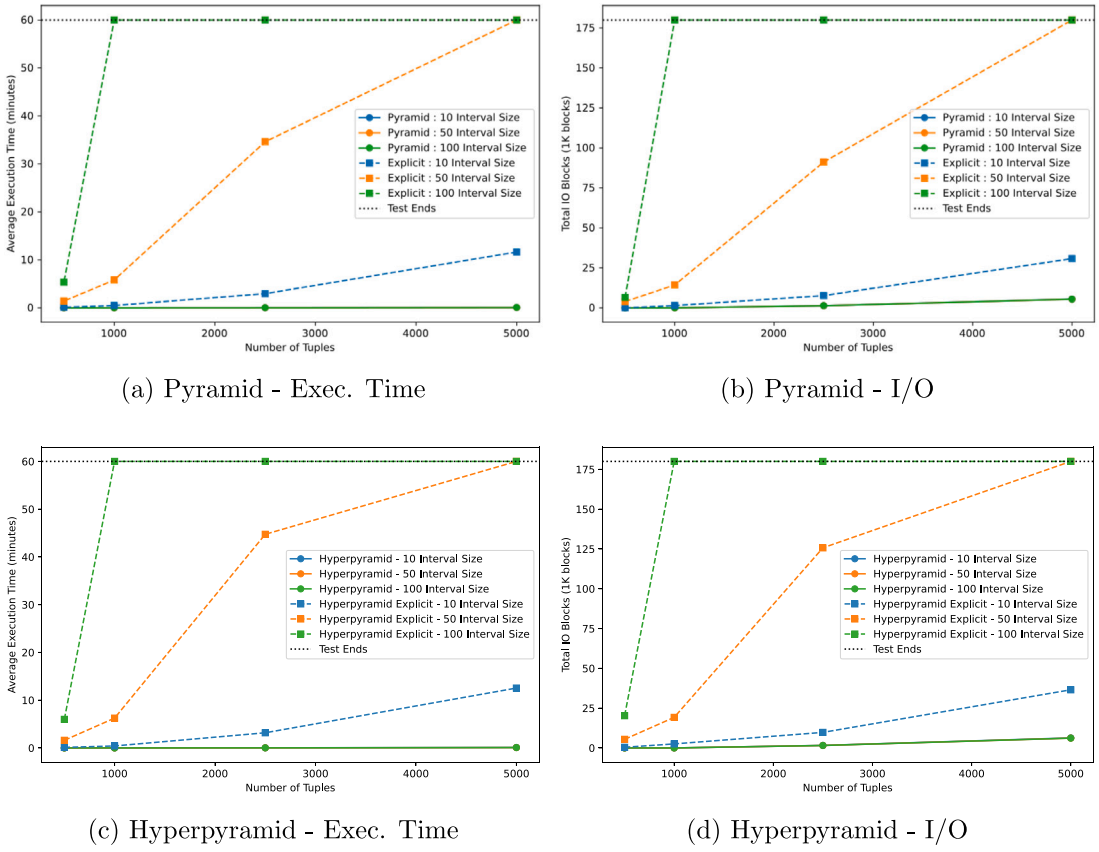
This study was partially funded by Fondazione Compagnia San Paolo and Fondazione Cassa Depositi e Prestiti (AI-LEAP project grant number D13C23001280007).

Appendix A. Auxiliary functions

In this Appendix we report the auxiliary functions used in the definition of the HP relational operators.

Cartesian product auxiliary operators:

$$V1_r \cap^{hp} V2_r = \langle V1_r(1) \cap^{cuboid} V2_r(1), \ldots, V1_r(r) \cap^{cuboid} V2_r(r) \rangle$$



(a) Pyramid - Exec. Time (b) Pyramid - I/O

(c) Hyperpyramid - Exec. Time (d) Hyperpyramid - I/O

Fig. 12. Interval Sizes (Cartesian product): comparisons between implicit and explicit approach for Pyramid and Hyperpyramid with varying number of interval sizes as regards execution time and I/O, considering datasets of different sizes.

$V1_r(i) \cap^{cuboid} V2_r(i) = (max(s1^i_m, s2^i_m) : min(s1^i_M, s2^i_M), max(e1^i_m, e2^i_m) : min(e1^i_M, e2^i_M), max(d1^i_m, d2^i_m) : min(d1^i_M, d2^i_M))^i$

The operator $\cap^{hp}$ performs the intersection of two hyperpyramids by applying the $\cap^{cuboid}$ operation at each one of the r preference levels. In turn, $\cap^{cuboid}$ performs the intersection between two cuboids.

Difference auxiliary operators:

$S -^{hp}_{n\_to\_m} \emptyset = S$

$S1 -^{hp}_{n\_to\_m} S2 = \bigcup_{V1'_r \in S1} V1'_r -^{hp}_{1\_to\_m} S2$

$V1'_r -^{hp}_{1\_to\_m} \{V2'_{r,1}, \ldots, V2'_{r,k}\} = (V1'_r -^{hp}_{1\_to\_1} V2'_{r,1}) -^{hp}_{n\_to\_m} \{V2'_{r,2}, \ldots, V2'_{r,k}\}$, where

$V1_r -^{hp}_{1\_to\_1} V2_r = \{V1_r -^{cuboid}_{left} V2_r, V1_r -^{cuboid}_{top} V2_r, V1_r -^{cuboid}_{right} V2_r, V1_r -^{cuboid}_{bottom} V2_r, V1_r -^{cuboid}_{front} V2_r, V1_r -^{cuboid}_{back} V2_r\}$, where

Let $V1_r = \langle (s1^1_m : s1^1_M, e1^1_m : e1^1_M, d1^1_m : d1^1_M)^1, \ldots, (s1^r_m : s1^r_M, e1^r_m : e1^r_M, d1^r_m : d1^r_M)^r \rangle$ and $V2_r = \langle (s2^1_m : s2^1_M, e2^1_m : e2^1_M, d2^1_m : d2^1_M)^1, \ldots, (s2^r_m : s2^r_M, e2^r_m : e2^r_M, d2^r_m : d2^r_M)^r \rangle$,

$V1_r -^{cuboid}_{left} V2_r = ((s1^i_m : max(s1^i_m, s2^1_m), min(e1^i_m, e2^1_m) : max(e1^i_M, e2^1_M), max(d1^i_m, d2^1_m) : d1^i_M)^i \ with \ 1 \le i \le r)$ **(A)**,

$V1_r -^{cuboid}_{top} V2_r = ((max(s1^i_m, s2^1_m) : s1^i_M, min(e1^i_m, e2^1_m) : max(e1^i_M, e2^1_M), min(d1^i_m, d2^1_M) : d1^i_M)^i \ with \ 1 \le i \le r)$ **(B)**,

$V1_r -^{cuboid}_{right} V2_r = ((min(s1^i_M, s2^1_M) : s1^i_M, min(e1^i_m, e2^1_m) : max(e1^i_M, e2^1_M), d1^i_m : min(d1^i_M, d2^1_M))^i \ with \ 1 \le i \le r)$ **(C)**,

$V1_r -^{cuboid}_{bottom} V2_r = ((min(s1^i_M, s2^1_M) : s1^i_M, min(e1^i_m, e2^1_m) : max(e1^i_M, e2^1_M), d1^i_m : min(d1^i_M, d2^1_M))^i \ with \ 1 \le i \le r)$ **(D)**,

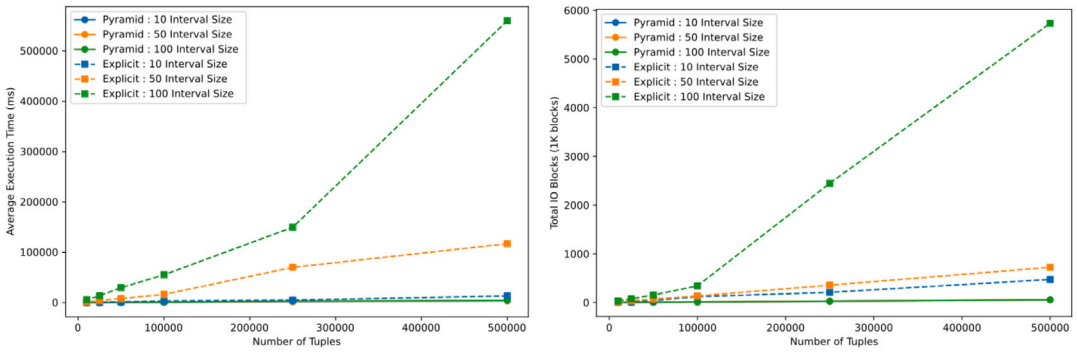
$V1_r -^{cuboid}_{front} V2_r = ((max(s1^i_m, s2^1_m) : min(s1^i_M, s2^1_M), min(e1^i_m, e2^1_m) : min(e1^i_M, e2^1_M), max(d1^i_m, d2^1_m) : min(d1^i_M, d2^1_M))^i \ with \ 1 \le i \le r)$ **(E)**,

$V1_r -^{cuboid}_{back} V2_r = ((max(s1^i_m, s2^1_m) : min(s1^i_M, s2^1_M), max(e1^i_m, e2^1_M) : max(e1^i_M, e2^1_M), max(d1^i_m, d2^1_m) : min(d1^i_M, d2^1_M))^i \ with \ 1 \le i \le r)$ **(F)**.

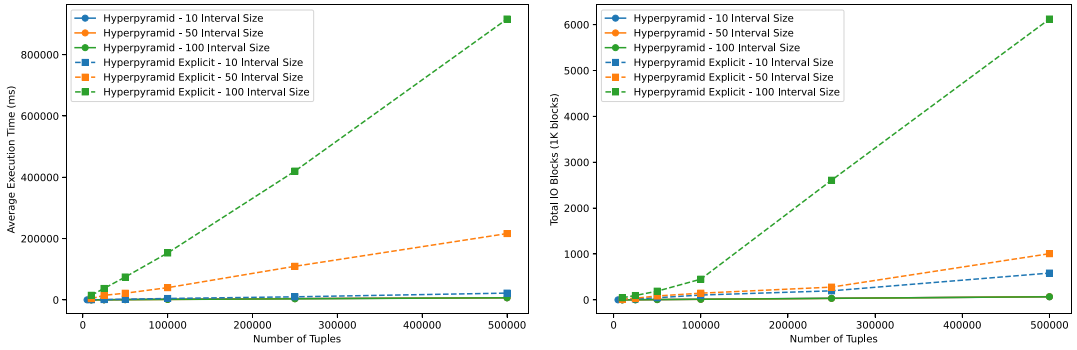
Relational difference between two hyperpyramids is performed through the $-^{hp}$ operations, that are defined in the cases many-to-many $-^{hp}_{n\_to\_m}$, one-to-many $-^{hp}_{1\_to\_m}$ and one-to-one $-^{hp}_{1\_to\_1}$. $-^{hp}_{1\_to\_1}$, in its turn, is defined by means of difference between preference levels, where each preference level of a hyperpyramid corresponds to a rectangular cuboid. As an example, in Fig. B.18 we show graphically the application of $-^{hp}_{1\_to\_1}$ at a given level of preference, in case the preference level $(s2^1_m : s2^1_M, e2^1_m : e2^1_M, d2^1_m : d2^1_M)^1$ of the subtrahend hyperpyramid is contained into the preference level $(s1^1_m : s1^1_M, e1^1_m : e1^1_M, d1^1_m : d1^1_M)^1$ of the minuend hyperpyramid (see Fig. B.18 (left)). At most six hyperpyramids may be generated as output of $-^{hp}_{1\_to\_1}$, corresponding to the parts labeled A, B, C, D, E, F in Definition 3.4 of difference (see Fig. B.18 (right)).

While $-^{hp}_{1\_to\_1}$ operates on two hyperpyramids, in case a tuple v in $r1^{HP}$ with hyperpyramid $V1_r$ (as its temporal extent) has multiple value-equivalent tuples $v_1, \ldots, v_k$ in $r2^{HP}$, the hyperpyramids of $v_1, \ldots, v_k$ must be deleted from $V1_r$. The operations $-^{hp}_{n\_to\_m}$ and $-^{hp}_{1\_to\_m}$ are recursively used to generalize $-^{hp}_{1\_to\_1}$ as an operation applied to sets of hyperpyramids (many-to-many and one-to-many applications of $-^{hp}_{1\_to\_1}$ respectively). It is worth noting that the application of many-to-many temporal difference between valid times is not specific of our approach but it is widely used in relational temporal DB approaches to cope with multiple value-equivalent tuples in the minuend (see, e.g., [1,23]).

Also notice that, given two hyperpyramids $V1_r$ and $V2_r$, $V1_r -^{hp}_{1\_to\_1} V2_r$ removes the bottom level $V2_r(1)$ of $V2_r$ from all the levels $V1_r(i)$ such that $1 \le i \le r$ of $V1_r$. The reason is that the presence of any level of preference for a triple $\langle s, e, d \rangle$ in the subtrahend models the fact that a tuple with (non-temporal) value v is present in the subtrahend at time $\langle s, e, d \rangle$, so that v at time $\langle s, e, d \rangle$ must not be part of the output. This is similar to other treatments of temporal indeterminacy (see, e.g., [20]).

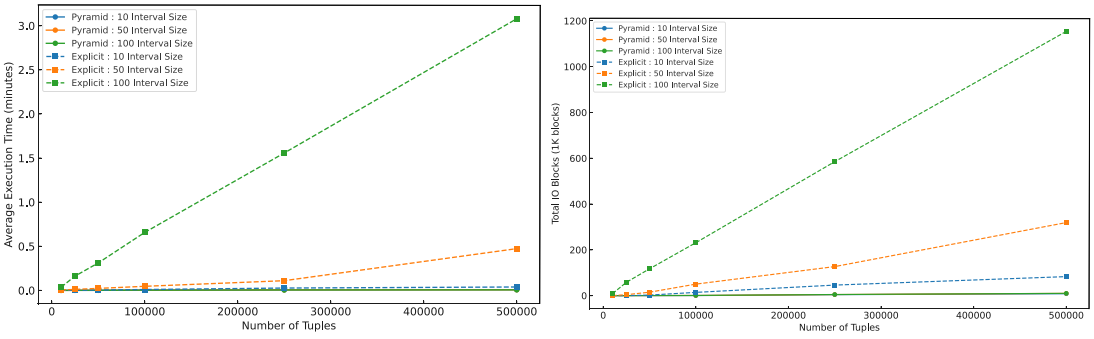


(a) Pyramid - Exec. Time (b) Pyramid - I/O

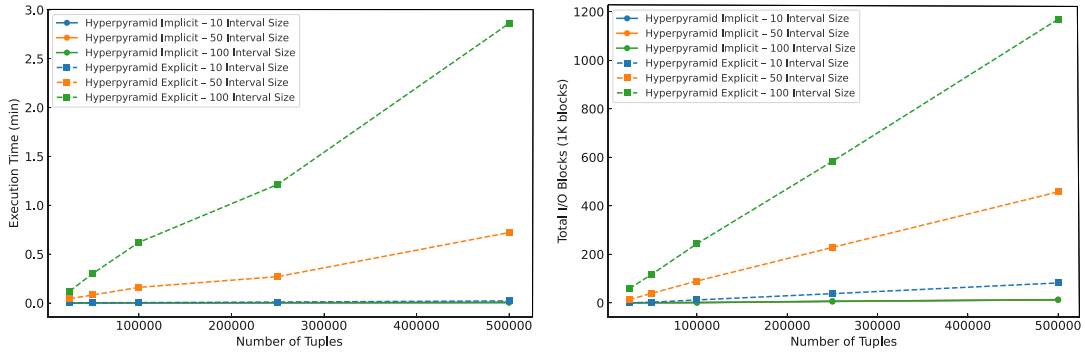


(c) Hyperpyramid - Exec. Time (d) Hyperpyramid levels - I/O

Fig. 13. Interval Sizes (difference): comparisons between implicit and explicit approach for Pyramid and Hyperpyramid with varying number of interval sizes as regards execution time and I/O, considering datasets of different sizes.



(a) Pyramid - Exec. Time (b) Pyramid - I/O



(c) Hyperpyramid - Exec. Time (d) Hyperpyramid - I/O

Fig. 14. Interval Sizes (temporal selection): comparisons between implicit and explicit approach for Pyramid and Hyperpyramid with varying number of interval sizes as regards execution time and I/O, considering datasets of different sizes.

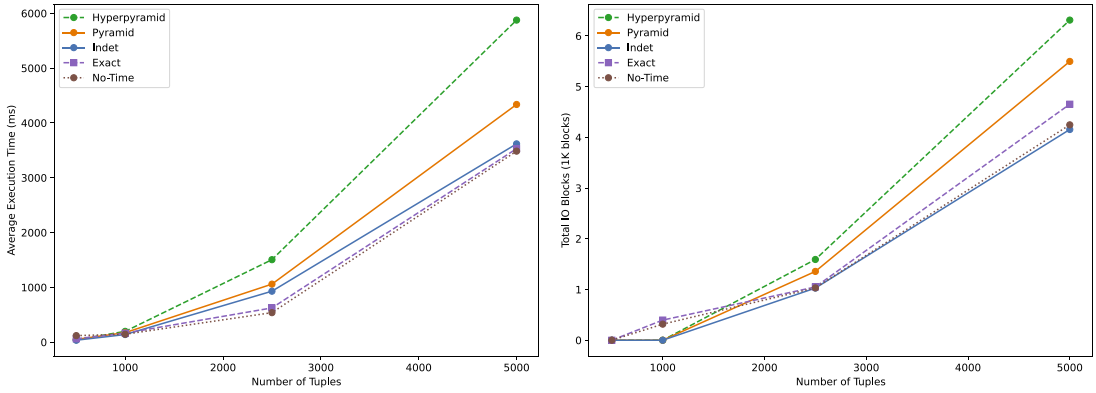


Fig. 15. Overhead (Cartesian product): comparisons among Hyperpyramid, Pyramid, Indet, Exact and No-Time with different preference levels as regards execution time and I/O, considering datasets of different sizes.

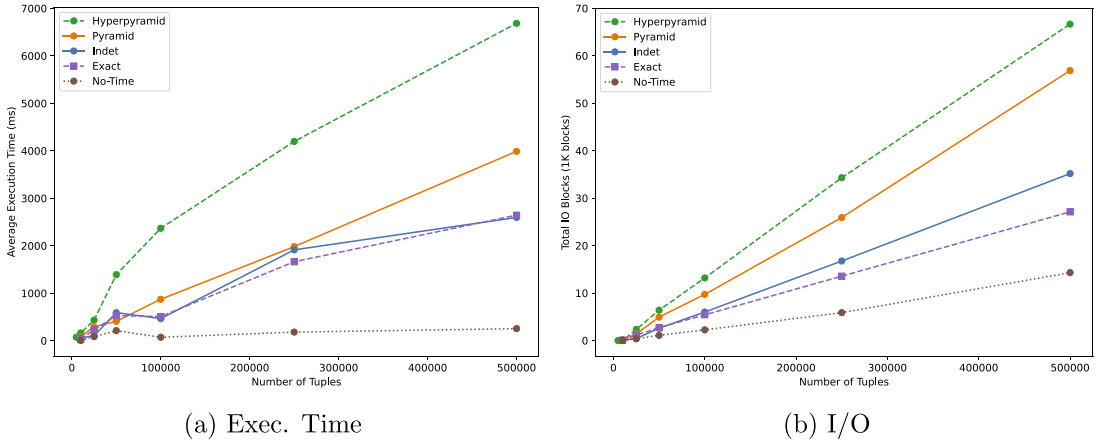


Fig. 16. Overhead (difference): comparisons among Hyperpyramid, Pyramid, Indet, Exact and No-Time with different preference levels as regards execution time and I/O, considering datasets of different sizes.

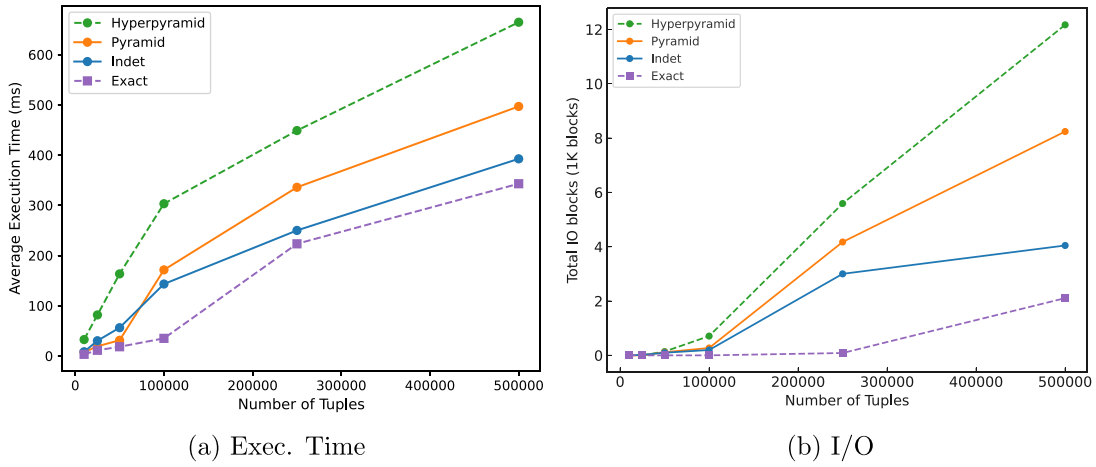


Fig. 17. Overhead (temporal selection): comparisons among Hyperpyramid, Pyramid, Indet and Exact with different preference levels as regards execution time and I/O, considering datasets of different sizes.

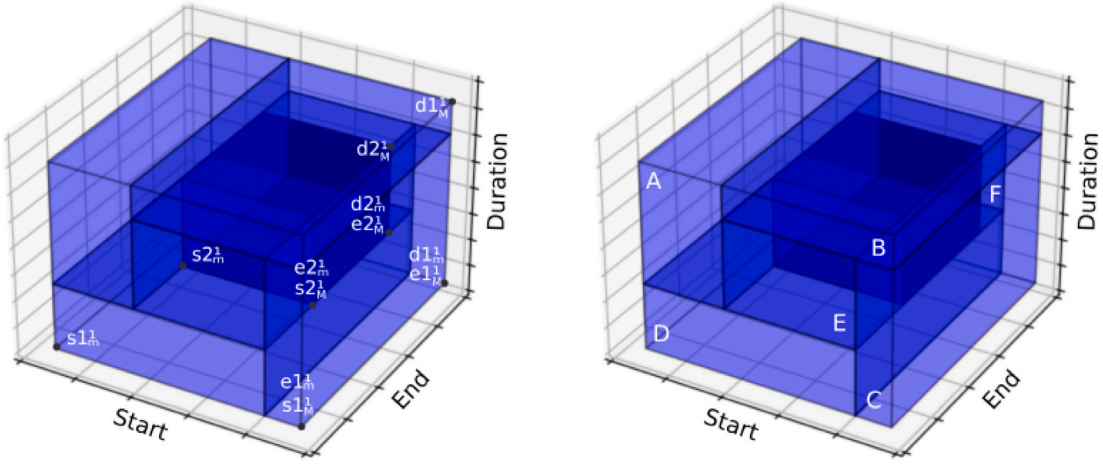


Fig. B.18. Hyperpyramid_diff at preference level 1: an example.

Appendix B. Proofs

Property B.1 (*Correctness of Time Interval Manipulation*)**.**

The temporal algebraic operators Op^{HP} extending Codd's operators are correct: for each Op^{HP}, the following property holds where Op^{Expl} is the algebraic operator on the explicit model corresponding to Op^{HP}:

$MakeExplicit(r1^{HP} Op^{HP} r2^{HP}) = MakeExplicit(r1^{HP})\ Op^{Expl}\ MakeExplicit(r2^{HP})$.

Proof. For the sake of brevity, we prove the property considering the Cartesian product, i.e., we prove that the following holds (the proofs for the other operators are similar):

$MakeExplicit(r1^{HP} \times^{HP} r2^{HP}) = MakeExplicit(r1^{HP}) \times^{Expl} MakeExplicit(r2^{HP})$.

Let $r1^{HP}$ and $r2^{HP}$ be two HP relations with schemas $(\overline{A}|\overline{T})$ and $(\overline{B}|\overline{T})$ respectively. We show the equivalence by proving the two inclusions separately, i.e., we prove that the left-hand side of the formula (henceforth lhs) implies the right-hand side (henceforth rhs) and vice versa.

($\mathbf{x'' \in lhs \Rightarrow x'' \in rhs}$)

Let $x''[\overline{A}, \overline{B}|\overline{S}] \in lhs$. Then, by definition of $MakeExplicit$ and Ext, there is a tuple $x'[\overline{A}, \overline{B}|\overline{T}] \in r1^{HP} \times^{HP} r2^{HP}$ such that $x'[\overline{A}, \overline{B}] = x''[\overline{A}, \overline{B}]$ and, for any $\langle s, e, d, p_i\rangle \in x''[\overline{S}]$ with $1 \le i \le r$, there is a V_r such that $x'[\overline{T}] = V_r$ and $V_r(i) = \langle s^i_m : s^i_M, e^i_m : e^i_M, d^i_m : d^i_M\rangle$, $\langle s, e, d\rangle \in Ext(V_r(i))$, i.e., $s^i_m \le s < s^i_M \wedge e^i_m \le e < e^i_M \wedge d^i_m \le d < d^i_M \wedge s + d = e$ and $p_i = S_r(i)$.

Given the definitions of $\times^{HP}$ and of $\cap^{hp}$, there exist two tuples $x1[\overline{A}|\overline{T}] \in r^{HP}$ and $x2[\overline{B}|\overline{T}] \in u^{HP}$ such that $x1[\overline{A}] = x'[\overline{A}]$, $x2[\overline{B}] = x'[\overline{B}]$ and, with $V_r = x'[\overline{T}]$, $V_r(i) = \langle s^i_m : s^i_M, e^i_m : e^i_M, d^i_m : d^i_M\rangle$, with $V1_r = x1[\overline{T}]$, $V1_r(i) = \langle s1^i_m : s1^i_M, e1^i_m : e1^i_M, d1^i_m : d1^i_M\rangle$ and with $V2_r = x2[T]$, $V2_r(i) = \langle s2^i_m : s2^i_M, e2^i_m : e2^i_M, d2^i_m : d2^i_M\rangle$, $\langle s^i_m : s^i_M, e^i_m : e^i_M, d^i_m : d^i_M\rangle = \langle max(s1^i_m, s2^i_m) : min(s1^i_M, s2^i_M), max(e1^i_m, e2^i_m) : min(e1^i_M, e2^i_M), max(d1^i_m, d2^i_m) : min(d1^i_M, d2^i_M)\rangle$.

Let us now consider again $\langle s, e, d, p_i\rangle \in x''[S]$. Since $s^i_m \le s < s^i_M$ and $s^i_m = max(s1^i_m, s2^i_m)$ and $s^i_M = min(s1^i_M, s2^i_M)$, then $max(s1^i_m, s2^i_m) \le s < min(s1^i_M, s2^i_M)$. Analogously, such properties also hold for d and e: since $d^i_m \le d < d^i_M$ and $d^i_m = max(d1^i_m, d2^i_m)$ and $d^i_M = min(d1^i_M, d2^i_M)$, then $max(d1^i_m, d2^i_m) \le d < min(d1^i_M, d2^i_M)$, and, since $e^i_m \le e < e^i_M$ and $e^i_m = max(e1^i_m, e2^i_m)$ and $e^i_M = min(e1^i_M, e2^i_M)$, then $max(e1^i_m, e2^i_m) \le e < min(e1^i_M, e2^i_M)$, where $s + d = e$.

Therefore, by definition of Ext, both $\langle s, e, d\rangle \in Ext(V1_r(i))$ and $\langle s, e, d\rangle \in Ext(V2_r(i))$. Thus, by definition of $MakeExplicit$, there is a tuple $x1'[\overline{A}|\overline{S}] \in MakeExplicit(r1^{HP})$ such that $x1'[\overline{A}] = x1[\overline{A}] = x'[\overline{A}]$ and $\langle s, e, d, p_i\rangle \in x1'[S]$, and there is a tuple $x2'[\overline{B}|\overline{S}] \in MakeExplicit(r2^{HP})$ such that $x2'[\overline{B}] = x2[\overline{B}] = x'[\overline{B}]$ and $\langle s, e, d, p_i\rangle \in x2'[S]$.

Thus, by definition of $\times^{Expl}$, there is a tuple $x12''[\overline{A}, \overline{B}|\overline{S}] \in rhs$ such that $x12''[\overline{A}] = x1'[\overline{A}]$, $x12''[\overline{B}] = x2'[\overline{B}]$ and, since both $\langle s, e, d, p_i\rangle \in x1'[\overline{S}]$ and $\langle s, e, d, p_i\rangle \in x2'[\overline{S}]$, $\langle s, e, d, p_i\rangle \in x1'[\overline{S}] \cap x2'[\overline{S}]$ and thus $\langle s, e, d, p_i\rangle \in x12''[\overline{S}]$.

By construction, $x12'' = x''$.

($\mathbf{x'' \in rhs \Rightarrow x'' \in lhs}$)

Let $x''[\overline{A}, \overline{B}|\overline{S}] \in rhs$. By definition of $\times^{Expl}$, there are tuples $x1'[\overline{A}|\overline{S}] \in MakeExplicit(r1^{HP})$ and $x2'[\overline{B}|\overline{S}] \in MakeExplicit(r2^{HP})$ such that $x1'[\overline{A}] = x''[\overline{A}]$, $x2'[\overline{B}] = x''[\overline{B}]$ and, for any $\langle s, e, d, p_i\rangle \in x''[\overline{S}]$, $\langle s, e, d, p_i\rangle \in x1'[\overline{S}]$ and $\langle s, e, d, p_i\rangle \in x2'[\overline{S}]$.

By definition of $MakeExplicit$ and of Ext, since $x1'[\overline{A}|\overline{S}] \in MakeExplicit(r1^{HP})$, there is a tuple $x1[\overline{A}|\overline{T}] \in r1^{HP}$ such that $x1'[\overline{A}] = x1[\overline{A}]$ and, with $V1_r = x1[\overline{T}]$, $\langle s, e, d\rangle \in Ext(V1_r(i))$. Analogously, since $x2'[\overline{B}|\overline{S}] \in MakeExplicit(r2^{HP})$, there is a tuple $x2[\overline{B}|\overline{T}] \in r2^{HP}$ such that $x2'[\overline{B}] = x2[\overline{B}]$ and, with $V2_r = x2[\overline{T}]$, $\langle s, e, d\rangle \in Ext(V2_r(i))$.

With $V1_r(i) = \langle s1^i_m : s1^i_M, e1^i_m : e1^i_M, d1^i_m : d1^i_M\rangle$, by definition of Ext, $s1^i_m \le s < s1^i_M$, $e1^i_m \le e < e1^i_M$, $d1^i_m \le d < d1^i_M$, and $s + d = e$.

Moreover, with $V2_r(i) = \langle s2^i_m : s2^i_M, e2^i_m : e2^i_M, d2^i_m : d2^i_M\rangle$, by definition of Ext, $s2^i_m \le s < s2^i_M$, $e2^i_m \le e < e2 + M^i$, $d2^i_m \le d < d2^i_M$, and, again, $s + d = e$.

Therefore, $max(s1^i_m, s2^i_m) \le s < min(s1^i_M, s2^i_M)$, $max(e1^i_m, e2^i_m) \le e < min(e1^i_M, e2^i_M)$ and $max(d1^i_m, d2^i_m) \le d < min(d1^i_M, d2^i_M)$. Then, by definition of $\times^{HP}$, there exists a tuple $x'[\overline{A}, \overline{B}|\overline{T}] \in r1^{HP} \times^{HP} r2^{HP}$ such that $x'[\overline{A}] = x1[\overline{A}]$, $x'[\overline{B}] = x2[\overline{B}]$, $x'[\overline{T}] = V_r$ with $V_r(i) = \langle max(s1^i_m, s2^i_m) : min(s1^i_M, s2^i_M), max(e1^i_m, e2^i_m) : min(e1^i_M, e2^i_M), max(d1^i_m, d2^i_m) : min(d1^i_M, d2^i_M)\rangle$ (and $max(s1^i_m, s2^i_m) \le s < min(s1^i_M, s2^i_M)$, $max(e1^i_m, e2^i_m) \le e < min(e1^i_M, e2_M i)$, $max(d1^i_m, d2^i_m) \le d < min(d1^i_M, d2^i_M)$) and $s + d = e$.

By definition of $MakeExplicit$, there is a tuple $x12''[\overline{A}, \overline{B}|\overline{S}]$ such that $x12''[\overline{A}, \overline{B}] = x'[\overline{A}, \overline{B}]$ and $\langle s, d, p_i\rangle \in x12''[\overline{S}]$.

By construction, $x12'' = x''$. ■

Property B.2 (*Reducibility to Codd's Algebra*)**.** *The HP relational algebra is reducible to Codd's relational algebra, i.e., for each pair of HP relations $r1^{HP}$ and $r2^{HP}$ defined over a proper schema, and considering a scale S_r, for each $s, e \in T^C, d \in \mathbb{N}$, and preference level i ($1 \le i \le r$) in the scale S_r the following holds:*

$\tau^i_{\langle s,e,d\rangle}(r1^{HP}\ Op^{HP}\ r2^{HP}) = \tau^i_{\langle s,e,d\rangle}(r1^{HP})\ Op^{Codd}\ \tau^i_{\langle s,e,d\rangle}(r2^{HP})$ *where Op^{HP} and Op^{Codd} represent corresponding relational operators in HP algebra and in Codd's algebra respectively.*

Proof. For brevity, we consider only Cartesian product, i.e., we prove $\tau^i_{\langle s,e,d\rangle}(r1^{HP} \times^{HP} r2^{HP}) = \tau^i_{\langle s,e,d\rangle}(r1^{HP}) \times^{Codd} \tau^i_{\langle s,e,d\rangle}(r2^{HP})$.

The proofs for the other operators are similar.

Let $r1^{HP}$ and $r2^{HP}$ be two HP relations with schemas $(\overline{A}|\overline{T})$ and $(\overline{B}|\overline{T})$ respectively. We show the equivalence by proving the two inclusions separately, i.e., we prove that the left-hand side of the formula (lhs) implies the right-hand side (rhs) and vice versa.

($\mathbf{x''} \in \mathbf{lhs} \Rightarrow \mathbf{x''} \in \mathbf{rhs}$)

Let $x''[\overline{A}, \overline{B}] \in lhs$. By definition of $\tau^i_{\langle s,e,d\rangle}$, there is a tuple $x'[\overline{A}, \overline{B}|\overline{T}] \in r1^{HP} \times^{HP} r2^{HP}$ such that $x'[\overline{A}, \overline{B}] = x''[\overline{A}, \overline{B}]$ and there is a V_r such that $x'[\overline{T}] = V_r$ and, with $V_r(i) = \langle s^i_m : s^i_M, e^i_m : e^i_M, d^i_m : d^i_M\rangle$, $\langle s,e,d\rangle \in Ext(V_r(i))$, i.e., $s^i_m \leq s < s^i_M \wedge e^i_m \leq e < e^i_M \wedge d^i_m \leq d < d^i_M \wedge s + d = e$.

By definition of $\times^{HP}$ and of $\cap^{hp}$, there are two tuples $x1[\overline{A}|\overline{T}] \in r1^{HP}$ and $x2[\overline{B}|\overline{T}] \in r2^{HP}$ such that $x1[\overline{A}] = x'[\overline{A}]$, $x2[\overline{B}] = x'[\overline{B}]$ and $\langle s^i_m : s^i_M, e^i_m : e^i_M, d^i_m : d^i_M\rangle = \langle max(s1_m1^i, s2^i_m) : min(s1^i_M, s2^i_M), max(e1^i_m, e2_m2^i) : min(e1^i_M, e2^i_M), max(d1^i_m, d2^i_m) : min(d1^i_M, d2^i_M)\rangle$, where $V1_r = x1[\overline{T}]$, $V1_r(i) = \langle s1^i_m : s1^i_M, e1^i_m : e1^i_M, d1^i_m : d1^i_M\rangle$, $V2_r = x2[\overline{T}]$, $V2_r(i) = \langle s2^i_m : s2^i_M, e2^i_m : e2^i_M, d2^i_m : d2^i_M\rangle$.

Since $max(s1^i_m, s2^i_m) = s^i_m \leq s < s^i_M = min(s1^i_M, s2^i_M)$, $max(e1^i_m, e2^i_m) = e^i_m \leq e < e^i_M = min(e1^i_M, e2^i_M)$, $max(d1^i_m, d2^i_m) = d^i_m \leq d < d^i_M = min(d1^i_M, d2^i_M)$ and $s + d = e$, by definition of Ext, both $\langle s,e,d\rangle \in Ext(V1_r(i))$ and $\langle s,e,d\rangle \in Ext(V2_r(i))$. Thus, by definition of $\tau^i_{\langle s,e,d\rangle}$, there is a tuple $x1' \in \tau^i_{\langle s,e,d\rangle}(r1^{HP})$ such that $x1'[\overline{A}] = x1[\overline{A}] = x'[\overline{A}]$, and there is a tuple $x2' \in \tau^i_{\langle s,e,d\rangle}(r2^{HP})$ such that $x2'[\overline{B}] = x2[\overline{B}] = x'[\overline{B}]$.

Thus, by definition of $\times^{Codd}$, there exists a tuple $x12'' \in rhs$ such that $x12''[\overline{A}] = x1'[\overline{A}]$ and $x12''[\overline{B}] = x2'[\overline{B}]$.

By construction, $x12'' = x''$.

($\mathbf{x''} \in \mathbf{rhs} \Rightarrow \mathbf{x''} \in \mathbf{lhs}$)

Let $x''[\overline{A}, \overline{B}] \in rhs$. By definition of $\times^{Codd}$, there are two tuples $x1'[\overline{A}] \in \tau^i_{\langle s,e,d\rangle}(r1^{HP})$ and $x2'[\overline{B}] \in \tau^i_{\langle s,e,d\rangle}(r2^{HP})$ such that $x1'[A] = x''[\overline{A}]$, $x2'[\overline{B}] = x''[\overline{B}]$.

By definition of $\tau^i_{\langle s,e,d\rangle}$, there is a tuple $x1[\overline{A}|\overline{T}] \in r^{HP}$ such that $x1'[\overline{A}] = x1[\overline{A}]$ and, with $V1_r = x1[\overline{T}]$, $\langle s,e,d\rangle \in Ext(V1_r(i))$, and there is a tuple $x2[\overline{B}|\overline{T}] \in u^{HP}$ such that $x2'[\overline{B}] = x2[\overline{B}]$ and, with $V2_r = x2[\overline{T}]$, $\langle s,e,d\rangle \in Ext(V2_r(i))$.

With $V1_r = x1[\overline{T}] = \langle s1^i_m : s1^i_M, e1^i_m : e1^i_M, d1^i_m : d1^i_M\rangle$ and $V2_r = x2[\overline{T}] = \langle s2^i_m : s2^i_M, e2^i_m : e2^i_M, d2^i_m : d2^i_M\rangle$, we thus have that $s1^i_m \leq s < s1^i_M$, $s2^i_m \leq s < s2^i_M$, $e1^i_m \leq e < e1^i_M$, $e2^i_m \leq e < e2^i_M$, $d1^i_m \leq d < d1^i_M$, $d2^i_m \leq d < d2^i_M$, and $s + d = e$. As a consequence, $max(s1^i_m, s2^i_m) \leq s < min(s1_Mi, s2^i_M)$, $max(e1^i_m, e2^i_m) \leq e < min(e1^i_M, e2^i_M)$ and $max(d1^i_m, d2^i_m) \leq d < min(d1^i_M, d2^i_M)$.

Thus, by definition of $\times^{HP}$, there is a tuple $x'[\overline{A}, \overline{B}|\overline{T}] \in r1^{HP} \times^{HP} r2^{HP}$ such that $x'[\overline{A}] = x1[\overline{A}]$, $x'[\overline{B}] = x2[\overline{B}]$, $x'[\overline{T}] = V'_r$ with $V'_r(i) = \langle max(s1^i_m, s2^i_m) : min(s1^i_M, s2^i_M), max(e1^i_m, e2^i_m) : min(e1^i_M, e2^i_M), max(d1^i_m, d2^i_m) : min(d1^i_M, d2^i_M)\rangle$ such that $max(s1^i_m, s2^i_m) \leq s < min(s1^i_M, s2^i_M)$, $max(e1^i_m, e2^i_m) \leq e < min(e1^i_M, e2^i_M)$, $max(d1^i_m, d2^i_m) \leq d < min(d1^i_M, d2^i_M)$, and $s + d = e$.

Therefore, by definition of $\tau^i_{\langle s,e,d\rangle}$, there is a tuple $x12''$ such that $x12''[\overline{A}, \overline{B}] = x'[\overline{A}, \overline{B}]$.

By construction, $x12'' = x''$. ■

Data availability

Data will be made available on request.

References

[1] Richard T. Snodgrass (Ed.), The TSQL2 Temporal Query Language, Kluwer, 1995.

[2] Yu Wu, Sushil Jajodia, Xiaoyang Sean Wang, Temporal database bibliography update, in: Opher Etzion, Sushil Jajodia, Suryanarayana M. Sripada (Eds.), Temporal Databases: Research and Practice. (the Book Grow Out of a Dagstuhl Seminar, June 23-27, 1997), in: Lecture Notes in Computer Science, 1399, Springer, 1997, pp. 338–366.

[3] E.F. Codd, Relational completeness of data base sublanguages, Res. Rep. / RJ / IBM / San Jose, Calif. RJ987 (1972).

[4] Johann Gamper, Matteo Ceccarello, Anton Dignös, What's new in temporal databases? in: Silvia Chiusano, Tania Cerquitelli, Robert Wrembel (Eds.), Advances in Databases and Information Systems - 26th European Conference, ADBIS 2022, Turin, Italy, September 5-8, 2022, Proceedings, in: Lecture Notes in Computer Science, vol. 13389, Springer, 2022, pp. 45–58.

[5] Ling Liu, M. Tamer Özsu (Eds.), Encyclopedia of Database Systems, second ed., Springer, 2018.

[6] Curtis E. Dyreson, Richard T. Snodgrass, Supporting valid-time indeterminacy, ACM Trans. Database Syst. 23 (1) (1998) 1–57.

[7] Alex Dekhtyar, Robert B. Ross, V.S. Subrahmanian, Probabilistic temporal databases, I: algebra, ACM Trans. Database Syst. 26 (1) (2001) 41–95.

[8] Luca Anselma, Paolo Terenziani, Richard T. Snodgrass, Valid-time indeterminacy in temporal relational databases: Semantics and representations, IEEE Trans. Knowl. Data Eng. 25 (12) (2013) 2880–2894.

[9] Luca Anselma, Luca Piovesan, Paolo Terenziani, Dealing with temporal indeterminacy in relational databases: An AI methodology, AI Commun. 32 (3) (2019) 207–221.

[10] Luca Anselma, Luca Piovesan, Abdul Sattar, Bela Stantic, Paolo Terenziani, A comprehensive approach to 'now' in temporal relational databases: Semantics and representation, IEEE Trans. Knowl. Data Eng. 28 (10) (2016) 2538–2551.

[11] Luca Anselma, Luca Piovesan, Paolo Terenziani, A 1NF temporal relational model and algebra coping with valid-time temporal indeterminacy, J. Intell. Inf. Syst. 47 (3) (2016) 345–374.

[12] Didier Dubois, Hélène Fargier, Henri Prade, Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty, Appl. Intell. 6 (4) (1996) 287–309.

[13] Francesca Rossi, Kristen Brent Venable, Toby Walsh, A Short Introduction to Preferences: Between AI and Social Choice, Morgan & Claypool Publishers, 2011.

[14] Jonathan Cohen, Keith Marzilli Ericson, David Laibson, John Myles White, Measuring time preferences, J. Econ. Lit. 58 (2) (2020) 299–347.

[15] Antonella Andolina, Marco Guazzone, Luca Piovesan, Paolo Terenziani, Temporal reasoning and query answering with preferences and probabilities for medical decision support, Expert Syst. Appl. 195 (2022) 116565.

[16] Marjon Van Der Pol, Deirdre Hennessy, Braden Manns, The role of time and risk preferences in adherence to physician advice on health behavior change, Eur. J. Heal. Econ. 18 (2017) 373–386.

[17] Alessio Bottrighi, Federica Grosso, Marco Ghiglione, Antonio Maconi, Stefano Nera, Luca Piovesan, Erica Raina, Annalisa Roveta, Paolo Terenziani, A symbolic AI approach to medical training, J. Med. Syst. 49 (1) (2025) 2.

[18] Antonella Andolina, Luca Arborio, Alessio Bottrighi, Antonio Maconi, Stefano Nera, Luca Piovesan, Erica Raina, Paolo Terenziani, AI-based medical education through computer-interpretable clinical guidelines: Project and first advances, in: 2025 5th Asia Conference on Information Engineering, ACIE, IEEE, 2025, pp. 148–153.

[19] Paolo Terenziani, Efrat German, Yuval Shahar, The temporal aspects of clinical guidelines, in: Annette ten Teije, Silvia Miksch, Peter J.F. Lucas (Eds.), Computer-Based Medical Guidelines and Protocols: A Primer and Current Trends, in: Studies in Health Technology and Informatics, vol. 139, IOS Press, 2008, pp. 81–100.

[20] Luca Anselma, Paolo Terenziani, Preferences in temporal relational databases, IEEE Access 12 (2024) 65418–65427.

[21] Christian S. Jensen, Richard T. Snodgrass, Semantics of time-varying information, Inf. Syst. 21 (4) (1996) 311–352.

[22] Luca Anselma, Alessandro Mazzei, Luca Piovesan, Paolo Terenziani, Reasoning and querying bounds on differences with layered preferences, Int. J. Intell. Syst. 36 (5) (2021) 1998–2035.

[23] Abdullah Uz Tansel, Temporal algebras, in: Ling Liu, M. Tamer Özsu (Eds.), Encyclopedia of Database Systems, Second Edition, Springer, 2018.

[24] Georg Duftschmid, Silvia Miksch, Walter Gall, Verification of temporal scheduling constraints in clinical practice guidelines, Artif. Intell. Med. 25 (2) (2002) 93–121.

[25] Annette ten Teije, Silvia Miksch, Peter J.F. Lucas (Eds.), Computer-based Medical Guidelines and Protocols: A Primer and Current Trends, Studies in Health Technology and Informatics, vol. 139, IOS Press, 2008.

[26] Mor Peleg, Computer-interpretable clinical guidelines: A methodological review, J. Biomed. Inform. 46 (4) (2013) 744–763.

[27] Vittorio Brusoni, Luca Console, Paolo Terenziani, Barbara Pernici, Qualitative and quantitative temporal constraints and relational databases: Theory, architecture, and applications, IEEE Trans. Knowl. Data Eng. 11 (6) (1999) 948–968.

[28] Manolis Koubarakis, Representation and querying in temporal databases: the power of temporal constraints, in: Proceedings of IEEE 9th International Conference on Data Engineering, IEEE, 1993, pp. 327–334.

[29] Luca Anselma, Antonella Coviello, Paolo Terenziani, Evaluating a temporal relational algebra supporting preferences in temporal relational databases, in: Joe Tekli, Johann Gamper, Richard Chbeir, Yannis Manolopoulos (Eds.), Advances in Databases and Information Systems - 28th European Conference, ADBIS 2024, Bayonne, France, August 28-31, 2024, Proceedings, in: Lecture Notes in Computer Science, vol. 14918, Springer, 2024, pp. 32–44.