What's New in Temporal Databases?

Johann Gamper(✉), Matteo Ceccarello, and Anton Dignös

Free University of Bozen-Bolzano, Dominikanerplatz/piazza Dominican 3, 39100 Bolzano, Italy
{johann.gamper,matteo.ceccarello,anton.dignoes}@unibz.it

Abstract. Temporal databases has been an active research area since many decades, ranging from research work on query processing, most dominantly on selection and join queries, to new directions in models and semantics, such as for instance temporal probabilistic or streaming data. At the same time more database vendors have been integrating temporal features into their systems, most notably, the temporal features of the SQL standard. In this paper, we summarize the latest research developments as presented in 30 research papers over the last five years in the context of temporal relational databases. Additionally, we also describe the developments of industrial database systems and vendors.

Keyword: Temporal databases

1 Introduction

Temporal databases is an active research area since several decades, with a renewed interest in recent years. The interest in temporal databases is driven by a variety of old and new applications that require to store and process temporal data, such as versioning of web documents [21], management of normative texts [27], air traffic monitoring and patient care [5], video surveillance [44], sales analysis [41], financial market analysis [25], and data warehousing and analytics [48], to name a few.

In temporal databases every fact is associated with one or more timestamps [7]. The timestamps are typically formed by either a time period or a set of time points, though other forms of timestamps exist, such as temporal elements. While time points are easier from a conceptual viewpoint, time periods are practically more relevant and allow for efficient implementations. The timestamps can represent different aspects of time, most importantly *valid time* [32] that indicates the validity of a fact in the real world (e.g., a contract that exists over a given period of time) and *transaction time* [31] that indicates the time when a tuple is/was stored in the database (e.g., a contract that was stored over

Supported by the Autonomous Province of Bozen-Bolzano with research call "Research Südtirol/Alto Adige 2019" (project Enabling Industrial-Strength, Open-Source Temporal Query Processing – ISTeP).

S. Chiusano et al. (Eds.): ADBIS 2022, LNCS 13389, pp. 45–58, 2022.
https://doi.org/10.1007/978-3-031-15740-0_5

a given period of time and later on updated or deleted from the database). When both aspects of time are present in a relation, we have a bitemporal relation [30].

Figure 1 shows two temporal relations **Mgr** and **Pro**. The timestamps are half-open intervals and represent the tuples' valid time. Relation **Mgr** records managers of departments, where **Dept** is the name of the department, **MName** is the name of the manager and **T** is the time period for which the person manages the department. Relation **Pro** records projects running in departments, where **PName** is the name of the project, **PDept** is the department which runs the project, and **T** is the time period over which the project runs.

Mgr

Dept	MName	T
AI	Tom	[01/2022, 06/2022)
AI	Sue	[06/2022, 01/2024)
DB	Ann	[10/2021, 01/2023)

Pro

PName	PDept	T
ExTAI	AI	[04/2022, 05/2023)
AIHuM	AI	[01/2023, 01/2024)
TauDB	DB	[10/2021, 06/2022)

Fig. 1. A temporal database with two temporal relations.

The most widely used semantics for temporal databases is known as *sequence semantics*, where temporal queries are defined using the concept of *snapshot reducibility* [15,35,49]. Snapshot reducibility views a temporal database as a sequence of snapshots and constrains a temporal operator applied to a temporal relation to produce, at a time point t, the same result as the corresponding non-temporal operator applied to the snapshot at t, i.e., all input tuples that are valid at t. For instance, the result of a temporal count aggregation is defined "point-wise" by the result of a non-temporal count aggregation. The aim of temporal databases is to facilitate such kind of operations in queries that would otherwise result in long, error prone, and inefficient SQL queries [47].

Figure 2 reports the result of two temporal queries on our example database from Fig. 1. In particular, the result of the temporal join between the two relations to retrieve for each project the responsible manager is shown in Fig. 2a. The temporal join is performed by joining two temporal relations according to overlapping timestamps. That is, the result tuples are timestamped with the intersection of the overlapping time periods. In the literature the step of finding overlapping pairs of tuples is also referred to as overlap or interval join. This result of the temporal join is consistent with the traditional (non-temporal) join performed at each snapshot of the data. For instance, the snapshot at time point 04/2022 for relation **Pro** contains two projects: ExTAI from the AI department and TauDB from the DB department. The snapshot for relation **Mgr** at the same time point contains two managers: Tom for the AI department and Ann for the DB department. Performing a non-temporal join on these two snapshots gives the same result as the snapshot at time point 04/2022 for the relation in Fig. 2a, i.e., ExTAI is managed by Tom and TauDB is managed by Ann.

PName	PDept	MName	T
ExTAI	AI	Tom	[04/2022, 06/2022)
ExTAI	AI	Sue	[06/2022, 05/2023)
AIHuM	AI	Sue	[01/2023, 01/2024)
TauDB	DB	Ann	[10/2021, 06/2022)

(a) Temporal Join

PDept	Count	T
AI	1	[04/2022, 01/2023)
AI	2	[01/2023, 05/2023)
AI	1	[05/2023, 01/2024)
DB	1	[10/2021, 06/2022)

(b) Temporal Aggregation

Fig. 2. Example operations *temporal join* and *aggregation*.

Figure 2b shows the result of a temporal aggregation that counts, for each department, the number of projects stored in relation **Pro**. Also in this case the result is defined according to the snapshots, and in the result we have one project in the AI department from 04/2022 to 01/2023, because each snapshot within this period contains exactly one project for department AI.

Past research on temporal databases has been focusing on various aspects of managing and processing temporal data, most notably on data models, SQL-based query languages, and efficient evaluation algorithms for query processing. Due to the ubiquity of temporal data and the need for processing such data, more recently also industry caught up with the topic, resulting in several extensions of commercial and Open Source database systems (e.g., IBM DB2, Oracle, Teradata, and PostgreSQL) with various degrees of support for temporal data. Finally, the major extension in the SQL:2011 standard was the support for the representation of temporal data [7,34].

In this paper, we review the newest developments in temporal relational databases as presented in 30 research papers over the last five years, which extend the state of the art as described in [7]. More specifically, Sect. 2 provides an overview of the works on temporal query processing, mostly focusing on selection and join queries. Section 3 provides the works that focus on new research directions in the area of temporal data models and semantics, followed by an overview on the newest developments in industrial systems in Sect. 4. Finally, Sect. 5 concludes the paper and provides interesting topics for future work that received scant attention in the last years.

2 Query Processing of Primitive Operators

In this section, we focus on recent advancements in query processing, mainly from an algorithmic point of view.

2.1 Temporal Selection

A classic query involving intervals is the *overlap* query, which retrieves all tuples whose timestamp overlaps with the query period. Christodoulou et al. [14] introduce HINT, an index addressing this kind of problem. It partitions the timeline

in a hierarchy of regular grids of geometrically increasing granularities. The addition of an auxiliary index of non-empty partitions allows to improve efficiency on skewed data.

A richer type of selection query are *range-duration* queries, first introduced by Behrend et al. [4]: matching intervals need to both overlap with the query and have a duration within a bound that is also defined in the query. Traditional index structures for intervals deal with only one aspect of intervals, and thus miss the opportunity to leverage the selectivity of queries on both dimensions.

In [4], the authors introduce Period-Index⋆ to explicitly support range-duration queries. The index partitions the timeline in *buckets* that will contain any interval they intersect. Then, intervals in a bucket are further partitioned in *levels* based on their duration, with the minimum duration in each level increasing geometrically. Finally, each level is further partitioned in the time domain in order to efficiently retrieve intervals of a given duration based on their start time. This approach is adaptive to the distribution of start times, but assumes a Zipf-like distribution of durations.

Recently, an index deemed RD-Index supporting range-duration queries has been introduced by Ceccarello et al. [13]. This index partitions tuples in a two-dimensional grid according to their start time and their duration by taking into account the distribution of both dimensions. This allows to adapt to the distribution of the data, providing better performance than the state of the art. Experiments show that this index performs better than the state of th e art also on *mixed* workloads, where some queries constrain only the duration, some constrain only the position on the timeline, and some constrain both.

2.2 Temporal Joins

Binary Joins. Temporal binary joins are joins between two relations where the join predicate requires that the interval timestamps of the tuples in the two relations overlap. A specialized data structure, called the Overlap Interval Inverted Index, is proposed by Luo et al. [37] to efficiently compute binary interval joins. The index uses the end points of intervals as anchor points and approximates the nesting structure of intervals by establishing relationships between these anchor points. This information is then used to prune unnecessary comparisons.

Interval joins for in-memory data have also been studied by Bouros et al. [10]. The paper proposes optimizations of the forward scan algorithm [11] and develops a parallel version, where a thread is responsible of sweeping the timeline, and then forward scans are executed in parallel.

Another adaptation of the forward scan algorithm, specifically tailored to *skewed* data, is proposed by Hellings and Wu [28]. This algorithm enriches the forward scan algorithm with an auxiliary data structure, termed stab-forests, which allows to skip portions of the input relations that are provably not part of the result.

An approach that does not necessarily involve the development of ad-hoc index structures is presented by Dignös et al. [16]: overlap joins are rewritten as the union of two range-joins. This rewriting enables the computation of overlap

joins using an efficient sort-merge based algorithm for range-joins that is on par with other state-of-the-art techniques and allows to efficiently implement overlap joins on widely available DBMS systems using B+-trees. Besides traditional overlap joins, this work also considers additional equality predicates in overlap joins as well as period boundaries that can have different interval definitions (e.g., closed or half-open) for different tuples.

Joins involving predicates on intervals can be extended in several directions. The most natural extension involves considering all of Allen's interval relations, as is done by Piatov et al. [46]. The paper takes the moves from [45], leveraging the *endpoint index* and the *gapless hash map* to efficiently process intervals in a cache-friendly way.

The *interval count semi-join* problem [9] requires instead to count for each interval of a relation R, the number of intervals in another relation S with which it overlaps. The paper extends the plane-sweep algorithm of [45] to solve this problem directly, without requiring a join followed by an aggregation step.

Finally, another extension is that of *band join* of intervals [8]. Specifically, the problem requires to join intervals that either overlap or whose smallest difference between endpoints is smaller than a parameter ϵ.

Multi-way Joins. In many cases, multiple temporal relations are to be joined. The traditional way of addressing this type of queries relies on finding the best sequence of binary joins. This approach has the drawback of potentially producing intermediate results which are much larger than the final output. In the past few years there has been a growing interest for multi-way equi-joins, following the development of the output optimal worst-case join [40], where the output of the join is computed by considering all involved relations at the same time.

Very recently, Hu et al. [29] developed an approach based on worst-case optimal join algorithms to deal with multi-way temporal joins. These algorithms are worst case optimal in the sense that, for a given query, one can bound the worst case output size based on the characteristics of the query: the algorithm will then run in time proportional to this worst case size. Furthermore, they introduce the problem of *durable* joins: only the intervals with duration longer than a given parameter τ are part of the output, allowing to ignore transient patterns.

The complexity of multi-way interval joins is studied by Khamis et al. [33]. Specifically, the paper provides a reduction of a multi-way interval join to a disjunction of multi-way equi-joins, and the corresponding backward reduction. This allows to both upper bound the complexity of multi-way interval joins and to state hardness results.

General intersection joins are the topic of the work by Tao and Yi [56]. Intersection joins consider overlaps between d-dimensional rectangles; for $d = 1$ the problem corresponds to overlap joins in temporal databases. The paper focuses on the *dynamic* variant of the problem, where one wants to update the solution as the relations involved in the join are modified. For binary joins of

intervals, the paper provides an optimal data structure requiring $O(n)$ space and $O(\log n)$ amortized update time.

Multi-way temporal joins also arise in the context of finding temporal subgraphs, like k-cliques of overlapping intervals [57].

3 New Directions in Models and Semantics

In this section, we provide an overview about new research directions which go beyond traditional temporal databases and include new semantics, data models, and query types.

Semantics. Most works in temporal databases focus on duplicate free temporal relations, i.e., set semantics, where value-equivalent tuples are not allowed to overlap. The work by Dignös et al. [17] provides the first theoretical foundations for processing temporal data with multiset semantics under full relational algebra and aggregation. In particular, this paper defines multiset semantics by adopting a novel data model based on the concepts of K-relations and semirings, which satisfy the properties of snapshot reducibility. The authors show how the temporal operators over temporal relations with multiset semantics can be translated into standard SQL queries via a query rewriting approach.

Implementing *sequenced semantics* using standard relational algebra is the goal of [19]. To this end, the paper proposes to use *log-segmented timestamps* [18] rather than time intervals. Assuming that the timeline has $n = 2^k$ chronons, labels of $b \leq k$ bits can be univocally associated to pre-determined time intervals. Therefore, any arbitrary time interval can be encoded using a collection of at most k labels. The paper proposes to transform a temporal relation in a non-temporal relation featuring labels in place of temporal intervals, and where each tuple is replicated up to k times, depending on the temporal interval to which it is associated. This transformation allows to express temporal queries using standard (non-temporal) relational algebra, and thus allowing to implement sequenced semantics in standard DBMSs without modifications.

Temporal Probabilistic Databases. A *temporal probabilistic database* [42] is a database complying with both the possible world semantics [51] of probabilistic databases and the sequenced semantics of temporal databases. In summary, a temporal probabilistic database can be thought of a collection of probabilistic databases, one for each time instant. The query semantics then requires that the result of any operation at any time point t is equivalent to the result derived from the corresponding probabilistic operation applied to the probabilistic database at time t. Set operations in this model are investigated by Papaioannou et al. [42], whereas [43] studies the problems of outer and anti joins.

Streaming Data. Nowadays many applications have to deal with incoming streams of data, rather than static datasets to be stored and processed offline.

The work by Suzanne et al. [52,53] considers the aggregation of spanning events (events with time periods) in the context of data streams. The work provides a framework that extends window aggregation over regular events with time points to spanning events with time periods. The framework supports a wide range of common window definitions for the aggregation, and it considers different ways how spanning events may be received in a data stream, e.g., an event may be received only at its end time, or an event is partially received first at its start time and later on completed at its end time. How window-slicing for spanning events can be performed to share computational costs among overlapping windows, is introduced in [54,55].

The work by Grandi et al. [26] proposes a query language and a unified algebraic framework that integrates streaming, temporal, and standard relational data in an all-in-one approach. This framework provides an extended relational algebra for one-time queries with temporal and non-temporal semantics as well as continuous queries with different types of window expressions, together with a translation that allows the execution of continuous queries using traditional temporal operators.

Ongoing Databases. The paper by Mülle and Böhlen [39] studies the concept of "now" [2,20] in temporal databases. While many approaches deal with time points declared as now by instantiating them to a given reference time (e.g., the current time), this solution provides a principled approach to deal with "now" during query processing, by keeping it uninstantiated and evaluating predicates and functions at all possible reference times. The result of a query is an ongoing relation that includes reference times. The authors introduce ongoing data types and their operations, a relational algebra for ongoing relations, and an implementation in PostgreSQL.

Historical What-If Queries. The work by Campbell et al. [12] introduces historical what-if queries that allow to determine the effect of hypothetical changes in the transactional history of a database. The approach exploits reenactment [3], a declarative replay technique for transactions, to simulate the evaluation of histories together with time travel [50] on transaction time to find the corresponding history of the data to apply the what-if scenario. The authors provide an optimization to apply historical changes only to the affected data together with an implementation as a middleware. While this approach does not focus on explicitly timestamped data, it exploits the transactional history of database systems.

Temporal Keyword Search. The work by Gao et al. [24] studies the problem of evaluating keyword queries with temporal predicates in temporal databases. The work shows how multiple interpretations and their corresponding SQL queries including temporal joins can be generated from the temporal predicates in the keyword search. The work in [23] shows how temporal aggregation and

span temporal aggregation [22] can be employed in temporal keyword search in order to allow users to query statistical information over time.

Data Warehouse. Ahmed et al. [1] show how to generalize and extend the multidimensional model used in data warehouses with temporal features and temporal online analytical processing (OLAP) operators. The work introduces a multidimensional model that is capable of representing traditional (non-temporal) and time-varying data independently with consistency constraints. The authors provide a mapping from the temporal model into the relational model and show how the temporal OLAP operators can be answered using standard SQL.

The work by Mahlknecht et al. [38] proposes different logical models how temporal data can be represented in a data warehouse to support efficient aggregations over time. The models differ in the way how data with time periods is stored: as the set of all time points in a time period, as the start and end time points of a time period, or as a combination of the two. The different models may or may not facilitate different aggregation operators over time that are frequently used in data warehouses. The authors show the queries in standard SQL and provide and experimental evaluation for the different models and aggregation operators on ETL performance and query time.

4 Systems

Database vendors have been gradually enhancing their database systems with support for temporal features, particularly with respect to the temporal features in the SQL:2011 standard [34]. After IBM DB2, Oracle DBMS, Teradata, and MS SQL Server, other database vendors have been following in the implementation of temporal features. In this review we focus on the new additions of the last five years and refer the reader to [6,7] for a more exhaustive study on the temporal features offered before.

MariaDB as of version 10.3.4 (Jan 2018) supports system-versioned tables[1] from the SQL:2011 standard, which provide integrated transaction time support. As of version 10.4.3 (Feb 2019), the database added also support for application-time period tables[2] (i.e., valid time relations). Since both temporal dimensions can be combined, MariaDB can also represent bitemporal tables. As of version 10.5.3 (May 2020), temporal uniqueness (WITHOUT OVERLAPS) was added, which can be used in the declaration of a table schema[3]. This feature allows to enforce temporal primary key constraints.

In a similar fashion, the in-memory, column-oriented relational database system SAP HANA introduced system-versioned tables as of version 2.0 SPS03[4]

[1] https://mariadb.com/kb/en/mariadb-1034-release-notes/.
[2] https://mariadb.com/kb/en/mariadb-1043-release-notes/.
[3] https://mariadb.com/kb/en/mariadb-1053-release-notes/.
[4] https://help.sap.com/doc/d25e2e530606453c9866c695298423b3/2.0.03/en-US/Whats_New_SAP_HANA_Platform_Release_Notes_en.pdf.

(Oct 2018), and application-time period tables as of version 2.0 SPS04[5] (Oct 2019).

The Open Source database system PostgreSQL followed a different route from the SQL:2011 standard. It offers support for temporal features through the build-in range types (period datatype) with associated operators and functions since version 9.2 (Sep 2012). As of version 14.0[6] (Sep 2021) PostgreSQL added support for multiranges[7] as a new data type, which are ordered lists of ranges with associated operators and functions.

The work by Lu et al. [36] provides a prototype built-in temporal implementation in Tencent's distributed database management system. The work integrates the features of the SQL:2011 standard into the system. It employs query rewriting in the parser to map queries on valid time into non-temporal queries. For transaction time several optimizations are proposed: a lazy migration strategy from the current to the history table that exploits the database management systems storage claiming procedure; a key/value store based approach to maintain only changed data instead of copies between current and history tables; and an optimized operator that retrieves current and historical data.

Other systems, such as CockroachDB and Snowflake, support time travel functionalities within a given retention time period. These systems allow to query and restore historical states of the data (if available). However, unlike in the SQL:2011 standard, versions and timestamps of the data are implicit and cannot be accessed.

5 Conclusion and Future Directions

In this paper, we reviewed new contributions in the field of temporal relational databases from the last five years. As a result, we survey 30 papers that span different areas in temporal relational databases: query processing with selection queries and joins, and new directions with topics such as improved temporal semantics, temporal probabilistic databases, streaming data, and more. Finally, we also summarized the newest developments with regards to temporal features in commercial and Open Source database systems.

While we have noticed that join algorithms have received most attention in the last few years, we also identified several topics that did not receive attention at all or are underrepresented. One such topic that requires deeper investigations is cost or cardinality estimation for temporal query operators. This is particularly important for temporal joins. As of today most query optimizers use heuristics or constants for the selectivity estimation of joins in the presence of inequalities. More precise cost estimation algorithms and their tight integration into query optimizers would be helpful to further improve the efficiency of query processing.

[5] https://help.sap.com/doc/d25e2e530606453c9866c695298423b3/2.0.04/en-US/Whats_New_SAP_HANA_Platform_Release_Notes_en.pdf.
[6] https://www.postgresql.org/docs/release/14.0/.
[7] https://www.postgresql.org/docs/current/functions-range.html.

Secondly, more research work on SQL extensions for temporal operators is needed, which is not covered in the SQL:2011 standard. While there exists some past research on this aspect, none of the proposed extensions received wide acceptance. The availability of a standard for the easy formulation of temporal queries in SQL may also help industry with the integration of temporal operators in their DBMS, in a similar fashion as the SQL:2011 standard pushed the development of temporal features.

Another direction for future research is concerned with query processing of bitemporal operators, which consider both valid time and transaction time. While the SQL:2011 standard allows to define and represent bitemporal tables, there exists only one work in the last five years that considers the computation of joins on more than one time dimension; all other works only focus on data that has either a valid time or a transaction time.

References

1. Ahmed, W., Zimányi, E., Vaisman, A.A., Wrembel, R.: A temporal multidimensional model and OLAP operators. Int. J. Data Warehous. Min. **16**(4), 112–143 (2020). https://doi.org/10.4018/IJDWM.2020100107
2. Anselma, L., Piovesan, L., Sattar, A., Stantic, B., Terenziani, P.: A comprehensive approach to 'now' in temporal relational databases: Semantics and representation. IEEE Trans. Knowl. Data Eng. **28**(10), 2538–2551 (2016). https://doi.org/10.1109/TKDE.2016.2588490
3. Arab, B.S., Gawlick, D., Krishnaswamy, V., Radhakrishnan, V., Glavic, B.: Reenactment for read-committed snapshot isolation. In: Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, 24–28 October 2016, pp. 841–850. ACM (2016). https://doi.org/10.1145/2983323.2983825
4. Behrend, A., Dignös, A., Gamper, J., Schmiegelt, P., Voigt, H., Rottmann, M., Kahl, K.: Period index: A learned 2d hash index for range and duration queries. In: Proceedings of the 16th International Symposium on Spatial and Temporal Databases, SSTD 2019, Vienna, Austria, 19–21 August 2019. pp. 100–109. ACM (2019). https://doi.org/10.1145/3340964.3340965
5. Behrend, A., Schmiegelt, P., Xie, J., Fehling, R., Ghoneimy, A., Liu, Z.H., Chan, E., Gawlick, D.: Temporal state management for supporting the real-time analysis of clinical data. In: Bassiliades, N., Ivanovic, M., Kon-Popovska, M., Manolopoulos, Y., Palpanas, T., Trajcevski, G., Vakali, A. (eds.) New Trends in Database and Information Systems II. AISC, vol. 312, pp. 159–170. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-10518-5_13
6. Böhlen, M.H., Dignös, A., Gamper, J., Jensen, C.S.: Database technology for processing temporal data (invited paper). In: 25th International Symposium on Temporal Representation and Reasoning, TIME 2018, Warsaw, Poland, October 15–17, 2018. LIPIcs, vol. 120, pp. 2:1–2:7. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018). https://doi.org/10.4230/LIPIcs.TIME.2018.2
7. Böhlen, M.H., Dignös, A., Gamper, J., Jensen, C.S.: Temporal data management – an overview. In: Zimányi, E. (ed.) eBISS 2017. LNBIP, vol. 324, pp. 51–83. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96655-7_3

8. Bouros, P., Lampropoulos, K., Tsitsigkos, D., Mamoulis, N., Terrovitis, M.: Band joins for interval data. In: Proceedings of the 23rd International Conference on Extending Database Technology, EDBT 2020, Copenhagen, Denmark, March 30–April 02, 2020, pp. 443–446. OpenProceedings.org (2020). https://doi.org/10.5441/002/edbt.2020.53
9. Bouros, P., Mamoulis, N.: Interval count semi-joins. In: Proceedings of the 21st International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, 26–29 March 2018, pp. 425–428. OpenProceedings.org (2018). https://doi.org/10.5441/002/edbt.2018.38
10. Bouros, P., Mamoulis, N., Tsitsigkos, D., Terrovitis, M.: In-memory interval joins. The VLDB J. **30**(4), 667–691 (2021). https://doi.org/10.1007/s00778-020-00639-0
11. Brinkhoff, T., Kriegel, H., Seeger, B.: Efficient processing of spatial joins using r-trees. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, 26–28 May 1993, pp. 237–246. ACM Press (1993). https://doi.org/10.1145/170035.170075
12. Campbell, F.S., Arab, B.S., Glavic, B.: Efficient answering of historical what-if queries. In: SIGMOD 2022: International Conference on Management of Data, Philadelphia, PA, USA, 12–17 June 2022, pp. 1556–1569. ACM (2022). https://doi.org/10.1145/3514221.3526138
13. Ceccarello, M., Dignös, A., Gamper, J., Khnaisser, C.: Indexing temporal relations for range-duration queries. CoRR abs/2206.07428 (2022). https://doi.org/10.48550/arXiv.2206.07428
14. Christodoulou, G., Bouros, P., Mamoulis, N.: HINT: A hierarchical index for intervals in main memory. In: SIGMOD 2022: International Conference on Management of Data, Philadelphia, PA, USA, 12–17 June 2022, pp. 1257–1270. ACM (2022). https://doi.org/10.1145/3514221.3517873
15. Dignös, A., Böhlen, M.H., Gamper, J., Jensen, C.S.: Extending the kernel of a relational DBMS with comprehensive support for sequenced temporal queries. ACM Trans. Database Syst. **41**(4), 26:1–26:46 (2016). https://doi.org/10.1145/2967608
16. Dignös, A., Böhlen, M.H., Gamper, J., Jensen, C.S., Moser, P.: Leveraging range joins for the computation of overlap joins. The VLDB J. **31**(1), 75–99 (2021). https://doi.org/10.1007/s00778-021-00692-3
17. Dignös, A., Glavic, B., Niu, X., Gamper, J., Böhlen, M.H.: Snapshot semantics for temporal multiset relations. Proc. VLDB Endow. **12**(6), 639–652 (2019). https://doi.org/10.14778/3311880.3311882
18. Dyreson, C.E.: Using CouchDB to compute temporal aggregates. In: 18th IEEE International Conference on High Performance Computing and Communications; 14th IEEE International Conference on Smart City; 2nd IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2016, Sydney, Australia, 12–14 December 2016, pp. 1131–1138. IEEE Computer Society (2016). https://doi.org/10.1109/HPCC-SmartCity-DSS.2016.0159
19. Dyreson, C.E., Ahsan, M.A.M.: Achieving a sequenced, relational query language with log-segmented timestamps. In: 28th International Symposium on Temporal Representation and Reasoning, TIME 2021, 27–29 September 2021, Klagenfurt, Austria. LIPIcs, vol. 206, pp. 14:1–14:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.TIME.2021.14
20. Dyreson, C.E., Jensen, C.S., Snodgrass, R.T.: Now in temporal databases. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, Second Edition. Springer, New York (2018). https://doi.org/10.1007/978-1-4614-8265-9_248

21. Dyreson, C.E., Lin, H., Wang, Y.: Managing versions of web documents in a transaction-time web server. In: Proceedings of the 13th International Conference on World Wide Web, WWW 2004, New York, NY, USA, 17–20 May 2004, pp. 422–432. ACM (2004). https://doi.org/10.1145/988672.988730
22. Gamper, J., Böhlen, M.H., Jensen, C.S.: Temporal aggregation. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, Second Edition. Springer, New York (2018). https://doi.org/10.1007/978-1-4614-8265-9_386
23. Gao, Q., Lee, M.L., Ling, T.W.: Temporal keyword search with aggregates and group-by. In: Ghose, A., Horkoff, J., Silva Souza, V.E., Parsons, J., Evermann, J. (eds.) ER 2021. LNCS, vol. 13011, pp. 160–175. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-89022-3_14
24. Gao, Q., Lee, M.L., Ling, T.W., Dobbie, G., Zeng, Z.: Analyzing temporal keyword queries for interactive search over temporal databases. In: Hartmann, S., Ma, H., Hameurlain, A., Pernul, G., Wagner, R.R. (eds.) DEXA 2018. LNCS, vol. 11029, pp. 355–371. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98809-2_22
25. Grandi, F., Mandreoli, F., Martoglia, R., Penzo, W.: A relational algebra for streaming tables living in a temporal database world. In: 24th International Symposium on Temporal Representation and Reasoning, TIME 2017, 16–18 October 2017, Mons, Belgium. LIPIcs, vol. 90, pp. 15:1–15:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017). https://doi.org/10.4230/LIPIcs.TIME.2017.15
26. Grandi, F., Mandreoli, F., Martoglia, R., Penzo, W.: Unleashing the power of querying streaming data in a temporal database world: A relational algebra approach. Inf. Syst. **103**, 101872 (2022). https://doi.org/10.1016/j.is.2021.101872
27. Grandi, F., Mandreoli, F., Tiberio, P.: Temporal modelling and management of normative documents in XML format. Data Knowl. Eng. **54**(3), 327–354 (2005). https://doi.org/10.1016/j.datak.2004.11.002
28. Hellings, J., Wu, Y.: Stab-forests: Dynamic data structures for efficient temporal query processing. In: 27th International Symposium on Temporal Representation and Reasoning, TIME 2020, 23–25 September 2020, Bozen-Bolzano, Italy. LIPIcs, vol. 178, pp. 18:1–18:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPIcs.TIME.2020.18
29. Hu, X., Sintos, S., Gao, J., Agarwal, P.K., Yang, J.: Computing complex temporal join queries efficiently. In: SIGMOD 2022: International Conference on Management of Data, Philadelphia, PA, USA, 12–17 June 2022, pp. 2076–2090. ACM (2022). https://doi.org/10.1145/3514221.3517893
30. Jensen, C.S., Snodgrass, R.T.: Bitemporal relation. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, pp. 243–244. Springer, New York (2009). https://doi.org/10.1007/978-0-387-39940-9_1409
31. Jensen, C.S., Snodgrass, R.T.: Transaction time. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, Second Edition, pp. 4200-4201. Springer, New York (2018). https://doi.org/10.1007/978-1-4614-8265-9_1064
32. Jensen, C.S., Snodgrass, R.T.: Valid time. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, Second Edition, pp. 4359–4360. Springer, New York (2018). https://doi.org/10.1007/978-1-4614-8265-9_1066
33. Khamis, M.A., Chichirim, G., Kormpa, A., Olteanu, D.: The complexity of Boolean conjunctive queries with intersection joins. In: PODS 2022: International Conference on Management of Data, Philadelphia, PA, USA, 12–17 June 2022, pp. 53–65. ACM (2022). https://doi.org/10.1145/3517804.3524156
34. Kulkarni, K.G., Michels, J.: Temporal features in SQL: 2011. SIGMOD Rec. **41**(3), 34–43 (2012). https://doi.org/10.1145/2380776.2380786

35. Lorentzos, N.A., Mitsopoulos, Y.G.: SQL extension for interval data. IEEE Trans. Knowl. Data Eng. **9**(3), 480–499 (1997). https://doi.org/10.1109/69.599935
36. Lu, W., Zhao, Z., Wang, X., Li, H., Zhang, Z., Shui, Z., Ye, S., Pan, A., Du, X.: A lightweight and efficient temporal database management system in TDSQL. Proc. VLDB Endow. **12**(12), 2035–2046 (2019). https://doi.org/10.14778/3352063.3352122
37. Luo, J.-Z., Shi, S.-F., Yang, G., Wang, H.-Z., Li, J.-Z.: O2iJoin: an efficient index-based algorithm for overlap interval join. J. Comput. Sci. Technol. **33**(5), 1023–1038 (2018). https://doi.org/10.1007/s11390-018-1872-x
38. Mahlknecht, G., Dignös, A., Kozmina, N.: Modeling and querying facts with period timestamps in data warehouses. Int. J. Appl. Math. Comput. Sci. **29**(1), 31–49 (2019). https://doi.org/10.2478/amcs-2019-0003
39. Mülle, Y., Böhlen, M.H.: Query results over ongoing databases that remain valid as time passes by. In: 36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, 20–24 April 2020, pp. 1429–1440. IEEE (2020). https://doi.org/10.1109/ICDE48307.2020.00127
40. Ngo, H.Q., Ré, C., Rudra, A.: Skew strikes back: new developments in the theory of join algorithms. SIGMOD Rec. **42**(4), 5–16 (2013). https://doi.org/10.1145/2590989.2590991
41. Papaioannou, K., Böhlen, M.H.: Temprora: top-k temporal-probabilistic results analysis. In: 32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, 16–20 May 2016, pp. 1382–1385. IEEE Computer Society (2016). https://doi.org/10.1109/ICDE.2016.7498350
42. Papaioannou, K., Theobald, M., Böhlen, M.H.: Supporting set operations in temporal-probabilistic databases. In: 34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, 16–19 April 2018, pp. 1180–1191. IEEE Computer Society (2018). https://doi.org/10.1109/ICDE.2018.00109
43. Papaioannou, K., Theobald, M., Böhlen, M.H.: Outer and anti joins in temporal-probabilistic databases. In: 35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, 8–11 April 2019, pp. 1742–1745. IEEE (2019). https://doi.org/10.1109/ICDE.2019.00187
44. Persia, F., Bettini, F., Helmer, S.: An interactive framework for video surveillance event detection and modeling. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, 06–10 November 2017, pp. 2515–2518. ACM (2017). https://doi.org/10.1145/3132847.3133164
45. Piatov, D., Helmer, S., Dignös, A.: An interval join optimized for modern hardware. In: 32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, 16–20 May 2016, pp. 1098–1109. IEEE Computer Society (2016). https://doi.org/10.1109/ICDE.2016.7498316
46. Piatov, D., Helmer, S., Dignös, A., Persia, F.: Cache-efficient sweeping-based interval joins for extended Allen relation predicates. The VLDB J. **30**(3), 379–402 (2021). https://doi.org/10.1007/s00778-020-00650-5
47. Snodgrass, R.T.: Developing Time-Oriented Database Applications in SQL. Morgan Kaufmann (1999)
48. Snodgrass, R.T.: A case study of temporal data. Teradata Corporation (2010)
49. Soo, M.D., Jensen, C.S., Snodgrass, R.T.: An algebra for TSQL2. In: The TSQL2 Temporal Query Language, chap. 27, pp. 501–544. Kluwer (1995)
50. Stonebraker, M.: The design of the POSTGRES storage system. In: VLDB 1987, Proceedings of 13th International Conference on Very Large Data Bases, September 1–4, 1987, Brighton, England, pp. 289–300. Morgan Kaufmann (1987)

51. Suciu, D., Olteanu, D., Ré, C., Koch, C.: Probabilistic Databases. Synthesis Lectures on Data Management. Morgan & Claypool Publishers (2011). https://doi.org/10.2200/S00362ED1V01Y201105DTM016
52. Suzanne, Aurélie, Raschia, Guillaume, Martinez, José: Temporal aggregation of spanning event stream: a general framework. In: Hartmann, Sven, Küng, Josef, Kotsis, Gabriele, Tjoa, A Min, Khalil, Ismail (eds.) DEXA 2020. LNCS, vol. 12392, pp. 385–395. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59051-2_26
53. Suzanne, A., Raschia, G., Martinez, J., Jaouen, R., Hervé, F.: Temporal aggregation of spanning event stream: an extended framework to handle the many stream models. Trans. Large Scale Data Knowl. Centered Syst. **49**, 1–32 (2021). https://doi.org/10.1007/978-3-662-64148-4_1
54. Suzanne, A., Raschia, G., Martinez, J., Tassetti, D.: Window-slicing techniques extended to spanning-event streams. In: 27th International Symposium on Temporal Representation and Reasoning, TIME 2020, 23–25 September 2020, Bozen-Bolzano, Italy. LIPIcs, vol. 178, pp. 10:1–10:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPIcs.TIME.2020.10
55. Suzanne, A., Raschia, G., Martinez, J., Tassetti, D.: Slicing techniques for temporal aggregation in spanning event streams. Inf. Comput. **281**, 104807 (2021). https://doi.org/10.1016/j.ic.2021.104807
56. Tao, Y., Yi, K.: Intersection joins under updates. J. Comput. Syst. Sci. **124**, 41–64 (2022). https://doi.org/10.1016/j.jcss.2021.09.004
57. Zhu, K., Fletcher, G.H.L., Yakovets, N., Papapetrou, O., Wu, Y.: Scalable temporal clique enumeration. In: Proceedings of the 16th International Symposium on Spatial and Temporal Databases, SSTD 2019, Vienna, Austria, 19–21 August 2019, pp. 120–129. ACM (2019). https://doi.org/10.1145/3340964.3340987