Encyclopedia of Information Science and Technology, Third Edition

Mehdi Khosrow-Pour
Information Resources Management Association, USA

A volume in the

Information Science REFERENCE
An Imprint of IGI Global

Managing Director: Lindsay Johnston
Production Editor: Jennifer Yoder & Christina Henning
Development Editor: Austin DeMarco & Jan Travers
Acquisitions Editor: Kayla Wolfe
Typesetter: Mike Brehm, John Crodian, Lisandro Gonzalez, Deanna Zombro
Cover Design: Jason Mull

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA, USA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Encyclopedia of information science and technology / Mehdi Khosrow-Pour, editor.
pages cm
Includes bibliographical references and index.
ISBN 978-1-4666-5888-2 (hardcover) -- ISBN 978-1-4666-5889-9 (ebook) -- ISBN 978-1-4666-5891-2 (print & perpetual access) 1. Information science--Encyclopedias. 2. Information technology--Encyclopedias. I. Khosrow-Pour, Mehdi, 1951-
Z1006.E566 2015
020.3--dc23

2014017131

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Temporal Databases

Fabio Grandi
University of Bologna, Italy

INTRODUCTION

Time is a ubiquitous aspect of real world phenomena and most computer applications require the management of time-varying information (e.g., processing of scientific and census data, banking and financial transactions, record-keeping and scheduling applications). Hence, the management of the temporal dimension has become a recognized important requirement of advanced database applications, in which the evolution of dynamic objects has to be represented in full relief and non-destructive changes must be applied to data. The advent of increasingly large and inexpensive storage devices has been the technological spring for the introduction of systems maintaining historical data and keeping track of past activities.

In this article we will briefly resume and discuss the main scientific results in the field of temporal databases. In particular, we will survey the features of proposed temporal data models based on extensions of the relational model and of temporal query languages based on extensions of the SQL standard, which are the most relevant for mainstream application development. Finally, we will survey the currently available implementations of temporal facilities in DBMS platforms.

BACKGROUND

Temporal databases have been an active research area for several decades, primarily focusing on temporal extensions of data models and query languages but also considering several other aspects of database technology. Extensions have been proposed mainly for the relational data model but also for object-oriented, XML, RDF and conceptual models like the Entity-Relationship model. A quite large literature, with pioneering works published in the early 1980s, is the outcome of such an effort as witnessed by several surveys and bibliographies (the latest thereof is Grandi (2012)), which also includes references to previous ones).

All extensions are based on the adoption of one or more time domains of interest for applications, whose values are used to assign a temporal pertinence (e.g., as *timestamps*) to data. The most popular and relevant time dimensions are *valid time* and *transaction time* (Jensen et al., 1998):

- The valid time of a fact is the time when the fact is true in the modeled reality.
- The transaction time of a fact is the time when the fact is stored in the database.

A database equipped with both valid and transaction time is said to be a *bitemporal* database. In a temporal database, a snapshot relation is a traditional relation, without time support. Further time dimensions (e.g., event/decision time, efficacy time or generic "user-defined" time) have also been considered in some application fields. For the modeling of a time domain, several aspects have been taken into account and studied, concerning its structure and features (e.g., discrete versus dense, linear versus branching, finite versus unbounded, besides granularity, periodicity, indeterminacy or probability, calendar support), and special values have been defined and characterized (e.g., "beginning," "now," "∞," "until changed"). In order to exploit the potentialities of a temporal database in applications, several temporal query languages (e.g., SQL extensions) have been proposed.

A milestone for the foundation and development of the discipline was the International Workshop on an Infrastructure for Temporal Databases, which was organized in 1993 under the auspices of the U.S. ARPA/NSF. As a side initiative, a panel of experts gathered to discuss and compile a consensus glossary of widely used technical terms specific to the temporal databases, which was published in 1994. A consolidated, revised and extended version of the glossary (Jensen et al.,

DOI: 10.4018/978-1-4666-5888-2.ch184

1998) came to light after the Dagstuhl Seminar on Temporal Databases held in 1997. Another follow-up of the ARPA/NSF workshop was the setting up of a committee, chaired by Richard Snodgrass, in charge of designing a temporal extension of the standard query language SQL-92: the TSQL2 Language Design Committee produced a first draft in 1994 and the final TSQL2 specification was published in a book (Snodgrass et al., 1995). Parts of TSQL2 were accepted by ANSI and included in a substandard of SQL3 named SQL/Temporal. Due to disagreements within the ISO committee, the project responsible for temporal support was canceled in 2001. However, concepts and constructs from SQL/Temporal were subsequently included in the latest SQL standard published in 2011 and have been implemented in several database platforms.

Temporal database studies have little by little entered into practice, as mainstream commercial DBMSs currently include some support for time-referenced data and temporal query facilities.

TEMPORAL EXTENSIONS OF THE RELATIONAL MODEL AND OF SQL

Temporal Data Models

Time can be associated with data in several different ways. In an extension of the relational model, time *points*, *intervals* or temporal *elements* (i.e., disjoint unions of intervals) can be used as timestamps. Temporal elements have been defined by Gadia (1988) in order to have a closed algebra of timestamp operators (differently from intervals, union and difference of elements is always an element). Moreover, *tuple-timestamping* or *attribute-timestamping* can be used (giving rise to also called *homogenous* and *inhomogeneous* models, respectively). In the former case, timestamps can be stored in implicit or explicit columns added to the table schema. In the latter case, a nested relation structure is needed to encode timestamping.

Let us consider, as simple example, the career of an employee which follows:

1. John was hired as a programmer (PRG) with initial salary 2K at time 1;
2. John's salary was raised to 3K at time 3 (but recorded in the DB at time 4);
3. John became a database administrator (DBA) at time 6.

Notice that (b) involves a retroactive update. This information can be stored in a valid-time, transaction-time or bitemporal table as shown in Figure 1 (tuple-timestamping with intervals is adopted; "–" means "until changed" or "forever" in valid time and "now" or "until changed" in transaction time). As it can also be verified from the figure, a valid-time relation allows users to effect retro- or pro-active changes, that is changes non necessarily effective when they are executed (but for which there is no way to know, after they were effected, whether they were on-time, retro- or pro-active). A transaction-time relation only allows to effect on-time changes, or it would be better to say that changes can only be interpreted as they were effective when applied (in our example, from Figure 1(2), it seems that John started earning 3K from time 4 and there is no way to see that arrears from time 3 were due). A bitemporal relation allows users to effect retro- and pro-active changes and to keep track of them (e.g., a tuple with valid start lesser than transaction start marks the result of a retroactive change). In the presence of retro- or pro-active changes, a bitemporal database only provides for full auditing and accountability.

The same data as in the valid-time relation of Figure 1(1) can also be represented as shown in Figure 2 by adopting other temporal modeling solutions. In particular, the representation in Figure 2(2), which is also representative of interval-timestamping employed at attribute level, corresponds to the case of a temporally *grouped* or history-oriented model (Clifford et al., 2005). In a grouped model, the temporal dimension is implicit in the structure of data representation and data objects are substituted by their histories: attributes can be regarded as functions that map time into domains (Gadia, 1988). Grouped models and query languages have been shown to be more expressive and friendly for human users. They do not lend themselves to implementation in a 1NF relational DBMS, but they have been proposed for implementation in nested relational or XML DBMSs (Wang, Zaniolo, & Zhou, 2005). Temporal models based on addition of timestamping columns are *ungrouped* indeed.

Another distinction involves *point-based* versus *interval-based* data models. In a point-based model, truth values of facts are associated to time points, whereas time intervals can be used merely as a compact

Figure 1. John's career data represented in a valid-time (1), transaction-time (2) and bitemporal (3) table

(1)

Name	Job	Salary	VT
John	PRG	2K	[1,2]
John	PRG	3K	[3,5]
John	DBA	3K	[6,–]

(2)

Name	Job	Salary	TT
John	PRG	2K	[1,3]
John	PRG	3K	[4,5]
John	DBA	3K	[6,–]

(3)

Name	Job	Salary	VT	TT
John	PRG	2K	[1,–]	[1,3]
John	PRG	2K	[1,2]	[4,–]
John	PRG	3K	[3,-]	[4,5]
John	PRG	3K	[3,5]	[6,–]
John	DBA	3K	[6,–]	[6,–]

Figure 2. The valid-time history of John's career represented via point-based tuple-timestamping (1) and by means of a temporally grouped model (2)

(1)

Name	Job	Salary	VT
John	PRG	2K	1
John	PRG	2K	2
John	PRG	3K	3
John	PRG	3K	4
John	PRG	3K	5
John	DBA	3K	6
John	DBA	3K	–

(2)

Name	Job	Salary
John	[1,5] → PRG	[1,2] → 2K
	[6,–] → DBA	[3,–] → 3K

representation or normalization tool (overlapping or adjacent timestamps of *value-equivalent* tuples can be merged via a *coalescence* operator into maximal intervals in order to produce a canonical representation). Querying in a point-based data model has the same expressive power of first-order temporal logic languages (Toman, 1996). In a strong interpretation of an interval-based model, interval timestamps are interpreted as indivisible as it is required for the so-called *telic* temporal data (Terenziani & Snodgrass, 2004). Telic facts, that represent accomplishments or achievements (like "the Golden Gate bridge was built from January 1933 to April 1937"), are true on an interval but false on any subinterval of it. On the contrary, *atelic* facts true on an interval are also true on any subinterval (e.g., "John worked as DBA in 2013"), lending themselves to be represented in a point-based data model. Temporal data of interest for management applications are usually atelic-type. In a weaker and more practical conception of interval-based model (Böhlen, Busatto, & Jensen, 1998), individuality of argument time intervals has to be preserved as much as possible by operators, as boundary points of an interval timestamp are reminiscent of their provenance from significant change events.

The adoption of temporal elements as timestamps is usually done in the framework of a point-based semantics. According to the approach of Jensen, Soo and Snodgrass (1994), a unifying data model called BCDM (Bitemporal Conceptual Data Model), based on temporal-element timestamping, can be defined at conceptual level and then mapped, at logical level, on different representational data models, which are designed with implementation in mind (e.g., based on interval-timestamping at tuple level, or on element-timestamping at attribute level indeed, rather than on backlogs, etc.). The structure of the chosen representational data model will reflect, at user interface level, on the required query language features (leading to the definition, according to Chomicki (1994), of a concrete temporal query language).

For temporal data models, temporal integrity constraints also involving notions of temporal key and referential integrity, temporal dependencies and normal forms have also been proposed.

Temporal Query Languages

If timestamps are added as explicit columns and, thus, can be treated in the same way as the other attributes at query language level, temporal data could be manipulated via standard SQL (Snodgrass, 1997). On the other hand, although not strictly necessary, special additional predicates and functions for manipulation of time (seen as a new abstract data type in SQL) can be provided as language extensions, basically to simplify the life of programmers. A further step is the full support of a period data type, which can be used for encoding interval-timestamps in a single column (endorsing an interval-based data model). If timestamps are added in an implicit way (i.e., they are inherent to the data model as in the case of TSQL2), special constructs have to be added to the query language and/or the semantics of some standard constructs has to be reconsidered in order to support temporal queries. The same applies to grouped data models.

Abstaining here from the subtle distinction between upward compatibility and temporal upward compatibility (Böhlen, Jensen, & Snodgrass, 2000), a temporal query language can be said to be *upward compatible* with SQL if non-temporal queries that can

be executed on the current snapshot of the database can also be executed on the temporal database and produce the same results. Upward compatibility is intended to guarantee a smooth migration of legacy applications and data to a temporal DBMS.

Some consistency notions for temporal query languages, or temporal algebraic operators on which query languages can be built upon, rely on the fact that a temporal relation can also be viewed as made up of a sequence of timestamped snapshot relations. Enforcement of the mutual consistency of the two viewpoints along the time axis leads to the notion of *snapshot reducibility*, that holds if each snapshot in the result of a temporal operator is equivalent to the result of the non-temporal counterpart of the same operator evaluated on the corresponding snapshot of the argument relation(s). Hence, if interval timestamping is adopted, the timestamps of the argument tuples are taken into account when forming the interval timestamps associated to the result tuples (e.g., interval intersection is used when executing a join). Enforcement of snapshot reducibility gives rise to a *sequenced semantics* in query execution.

However, snapshot reducibility does not apply to queries involving predicates and functions over the timestamps of argument relations, where snapshots valid at different times have to be mixed in their evaluation, which provide full temporal expressivity to a query language (e.g., to retrieve employees who were programmer before becoming DBA, which requires a *non-sequenced semantics* to evaluate). Moreover, preservation of the individuality of argument time intervals (i.e., to respect the weak form of interval-based semantics) has also been individuated as a desirable property of queries. In order to characterize the correctness of such kinds of queries too, the concepts of *extended snapshot reducibility* and *change preservation*, respectively, of query languages have been introduced (Dignös, Böhlen, & Gamper, 2012). Enforcement of extended snapshot reducibility by means of timestamp propagation (consisting in copying timestamp values to additional columns to be dealt with as non-temporal attributes) allows non-sequenced queries to be executed with a sequenced semantics. Enforcement of change propagation corresponds to the most correct, respectful of provenance, application of the sequenced semantics to data with true interval-timestamping.

In TSQL2, syntactic defaults have been embedded in the language specification in order to make the formulation of common temporal queries easier. For instance, intersection of the valid times of all the relations involved in a query to be assigned as timestamp to the results is automatically effected, yielding snapshot reducibility and enforcing a sequenced semantics by default (which, however, can be overridden by making timestamps explicit via direct reference to their values in expressions, or with a custom temporal projection specification). The period type and also temporal elements are fully supported, with a point-based perspective. Temporal selection is supported by means of predicates for (also mixed) time point, period and element comparison to be employed in the WHERE clause (namely OVERLAPS, =, CONTAINS, PRECEDES and MEETS) and of ancillary functions and constructors for management of time values.

An outstanding feature of TSQL2 is the availability of an implicit grouping mechanism with automatic timestamp coalescence for range variables declared in the FROM clause. For instance, the (non-sequenced) query which follows uses range variable declarations to conveniently retrieve the name of all the employees who changed job without a salary increase, together with the date of such a change:

```
SELECT SNAPSHOT Emp.Name,
BEGIN(VALID(Job2))
FROM Employee(Name) AS Emp,
Emp(Job,Salary)(PERIOD) AS Job1, Job2
WHERE Job1.Job <> Job2.Job
  AND Job1.Salary >= Job2. Salary
  AND VALID(Job1) MEETS VALID(Job2)
```

Range variables Job1 and Job2 bind to maximal groups of consecutive tuples in the history of the same employees, having a common value of Job and Salary attributes. The selection predicates force Job1 and Job2 to represent consecutive interval "versions" with different jobs and non increasing salary within such histories. The SNAPSHOT keyword in the target list forces implicit timestamps to be projected out from the returned tuples. If tuples are grouped on the time-invariant key of the relation (as for Emp in the example), this corresponds to superimpose a temporally grouped view over stored data (Clifford et al., 1995), with the possibility of declaring *history variables* and denoting versions within such histories. However, TSQL2 generalizes the grouping mechanism to arbitrary sets

of attributes and nesting levels, enabling an easy writing of very complex and powerful queries (e.g., involving the so-called *restructuring* (Gadia, 1986) of a temporal relation).

Moreover, TSQL2 was also designed to support event tables, temporal aggregates, multiple calendars and calendric systems, temporal indeterminacy, multiple temporal granularities, schema versioning and vacuuming (Snodgrass et al., 1995).

The direct successor of TSQL2, ATSQL (Böhlen, Jensen, & Snodgrass, 2000), introduces *statement modifiers* to override the TSQL2 defaults and make crystal clear the usage of a sequenced or non-sequenced semantics of execution, according to the explicit user specification.

The main criticism moved against TSQL2 and its successors, ATSQL and SQL/Temporal, proposed as SQL standard extension (Darwen & Date, 2006) concerned the adoption of hidden timestamp columns and the usage of the statement modifiers. A temporal SQL3 counterproposal submitted to ISO in 1995 was based on the IXSQL language (Lorentzos & Mitsopoulos, 1997). IXSQL supports a generic interval data type, which can be used for adding timestamps to temporal tables. Normalization of timestamps is enforced by means of two functions: *fold* and *unfold*. In order to execute a temporal query, unfold can be used to split interval-timestamped tuples into value-equivalent sets of point-timestamped tuples (e.g., converting the table in Figure 1(1) to the table in Figure 2(1)), before non-temporal operators can be applied, without distinction between non-temporal and time attributes, to execute the query. After the execution, fold is eventually used to merge timestamps of value-equivalent tuples into maximal intervals. Although IXSQL is interval-based in nature, when using the unfold operator, the underlying data model becomes point-based and change preservation does not hold, but extended snapshot reducibility can be supported. Application of a sequenced or non-sequenced semantics depends on the user's query.

Temporal Facilities in the SQL Standard and Commercial DBMSs

One of the main features of the new SQL:2011 (ISO/IEC 9075:2011) standard is improved support for temporal databases. In SQL:2011 (Kulkarni & Michels, 2011), transaction time support is granted by "system-versioned period tables" and valid time support is granted by "application-time period tables" (so that "system-versioned application-time period tables" are bitemporal tables). As far as the underlying temporal data model is concerned, the main difference with TSQL2 is that there is no new data type for time intervals (or elements) used as hidden dimensional column but two explicit date or timestamp columns can be declared together as tuple interval timestamp with a PERIOD FOR construct. Temporal keys and referential integrity constraints are supported. For instance, the schema of a table the most similar to the one in Figure 1(1) can be declared in SQL:2011 as follows:

```
CREATE TABLE Employee(
   Name VARCHAR(20), Job VARCHAR(20),
Salary INTEGER,
   VTStart DATE, VTEnd DATE, PERIOD
FOR VT (VTStart, VTEnd),
   PRIMARY KEY (Name, VT WITHOUT
OVERLAPS))
```

For querying, temporal predicates for interval comparison quite similar to the TSQL2 ones (namely OVERLAPS, CONTAINS, PRECEDES, SUCCEEDS, IMMEDIATELY PRECEDES and IMMEDIATELY SUCCEEDS) are provided. A validity can be specified by users in UPDATE and DELETE statements (e.g., to specify retro- or pro-active modifications).

As far as implementation of temporal database functionalities in relational DBMSs on the market is concerned, as of November 2013, we can compile the list that follows.

- IBM DB2, since the version 10 for z/OS released in 2010, includes built-in support of the period data type, valid and transaction time, upward compatibility, temporal primary keys, sequenced queries, in a manner very similar to that proposed for SQL/Temporal. From the latest DB2 version 10 for Linux, Unix and Windows released in 2012, adherence to the SQL:2011 standard has been introduced ("application time" has been dubbed as "business time").

- Oracle has provided support for temporal applications from the version 9i of 2001. In the latest version 12c, Oracle Workspace Manager enables application developers and DBAs to manage temporal data, with support of the period data type, valid-time, transaction-time and bitemporal tables, temporal primary keys, uniqueness and referential integrity constraints, sequenced and non-sequenced queries and updates, in a manner quite similar to that proposed for TSQL2 and SQL/Temporal.
- PostgreSQL has a couple of open-source contributed packages that can be installed in the database to manage temporal data: Temporal PostgreSQL and Temporal Tables Extension. The former basically adds temporal data types, functions and operators, whereas the latter adds transaction-time support in its current version, but is part of a project that aims at providing a full SQL:2011-compliant bitemporal extension.
- TimeDB is a free temporal relational DBMS by TimeConsult, implemented from the version 2.0 as a Java API designed to run as a frontend for Oracle. TSQL2/ATSQL statements (queries, updates, and assertions) are compiled into (sequences of) SQL-92 statements which are executed by the backend via JDBC, which makes it also compatible with other commercial platforms including IBM-Cloudscape and Sybase. TimeDB guarantees upward compatibility and provides support of bitemporal tables.
- Teradata Database, from the version 13.10 released in 2010, includes support of the period data type, valid and transaction time, temporal upward compatibility, temporal primary keys and referential integrity constraints, non sequenced and sequenced queries, in a manner similar to that proposed for TSQL2 (almost identical to that proposed for SQL/Temporal).

Support of bitemporal data is also provided in the Bloomberg-PolarLake Data Management Platform, or can be added to a DB without native support through the adoption of the Asserted Versioning Framework middleware or the usage of the Anchor Modeling database design tool.

Furthermore, some vendors provide tools that, in practice, can be used to add transaction-time temporal database facilities to leading DBMSs by working on their log files. These include IBM-SQL Replication for DB2, Lumigent-LogExplorer and ApexSQL-Log for Microsoft SQL Server, ASG-Time Navigator for most mainstream DBMSs as DB2, MySQL, Oracle, PostgreSQL, SQL Server and Sybase.

FUTURE RESEARCH DIRECTIONS

Future research should follow the direction of a seamless extension of temporal data base and query language solutions with advanced aspects still lacking in current standards and implementations, including, for instance, support of temporal elements, history-oriented (temporally grouped) view on data and change preservation. Implementation of temporal database functionalities and their integration with industrial-strength DBMS technology have not reached their maturity yet, and further research and development efforts in this direction are required in the next years.

CONCLUSION

In this article, we reviewed some of the most prominent theoretical results and practical achievements of research in the temporal database field. Although extensions required to any components of database technology have been studied, we focused on proposed temporal extensions of the relational model and of the SQL standard, discussed their main features and surveyed their emerging implementations in currents DBMSs. The considered aspects are the most relevant from the viewpoint of database application development. In particular, we showed that state-of-the-art temporal database solutions are often heritage of different research proposals (e.g., either TSQL2-like and IXSQL-like features can be found in commercial systems), in the quest for the best trade-off between theoretical soundness, user friendliness and implementation efficiency.

REFERENCES

Böhlen, M. H., Busatto, R., & Jensen, C. S. (1998). Point- Versus Interval-based Temporal Data Models. In S.D. Urban & E. Bertino (Eds.), *International Conference on Data Engineering* (pp. 192–200). Los Alamitos, CA: IEEE Computer Society Press.

Böhlen, M. H., Jensen, C. S., & Snodgrass, R. T. (2000). Temporal statement modifiers. *ACM Transactions on Database Systems*, *25*(4), 407–456. doi:10.1145/377674.377665

Chomicki, J. (1994). Temporal query languages: A survey. In D.M Gabbay & H.J. Ohlbach (Eds.), *International Conference* on *Temporal Logic* (pp. 506–534). Berlin, Germany: Springer.

Clifford, J., Croker, A., Grandi, F., & Tuzhilin, A. (1995). On Temporal Grouping. In J. Clifford, & A. Tuzhilin (Eds.), *Recent Advances in Temporal Databases* (pp. 194–213). Berlin, Germany: Springer. doi:10.1007/978-1-4471-3033-8_11

Darwen, H., & Date, C. J. (2006). An Overview and Analysis of Proposals Based on the TSQL2 Approach. In C. J. Date (Ed.), *Date on Databases: Writings 2000-2006* (pp. 481–514). New York, NY: Apress Media.

Dignös, A., Böhlen, M. H., & Gamper, J. (2012). Temporal Alignment. In K.S. Candan, Y. Chen, R.T. Snodgrass, L. Gravano & A. Fuxman (Eds.), *ACM SIGMOD International Conference on Management of Data* (pp. 433-444). New York, NY: ACM Press.

Gadia, S. K. (1986). Weak Temporal Relations. In A. Silberschatz (Ed.), *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (pp. 70–77). New York, NY: ACM Press.

Gadia, S. K. (1988). A Homogeneous Relational Model and Query Languages for Temporal Databases. *ACM Transactions on Database Systems*, *13*(4), 418–448. doi:10.1145/49346.50065

Grandi, F. (2012). Introducing an annotated bibliography on temporal and evolution aspects in the Semantic Web. *SIGMOD Record*, *41*(4), 18–21. doi:10.1145/2430456.2430460

Jensen, C. S., & Dyreson, C. E. (Eds.). Böhlen, M.H., Clifford, J., Elmasri, R., Gadia, S.K., … & Wiederhold, G. (1998). The consensus glossary of temporal database concepts – February 1998 version. In D. Etzion, S. Jajodla and S. Sripada (Eds.), Temporal Databases – Research and Practice (pp. 367–405). Berlin, Germany: Springer.

Jensen, C. S., Soo, M. D., & Snodgrass, R. T. (1994). Unifying Temporal Data Models via a Conceptual Model. *Information Systems*, *19*(7), 513–547. doi:10.1016/0306-4379(94)90013-2

Kulkarni, K., & Michels, J.-K. (2011). Temporal features in SQL:2011. *SIGMOD Record*, *41*(3), 34–43. doi:10.1145/2380776.2380786

Lorentzos, N. A., & Mitsopoulos, Y. G. (1997). SQL Extension for Interval Data. *IEEE Transactions on Knowledge and Data Engineering*, *9*(3), 480–499. doi:10.1109/69.599935

Snodgrass, R. T. (Ed.). Ahn, I., Ariav, G., Batory, D., Clifford, J., Dyreson, C.E., … & Sripada, S.M. (1995). The TSQL2 Temporal Query Language. Norwell, MA: Kluwer Academic Publishers.

Snodgrass, R. T. (1997). *Developing Time-Oriented Database Applications in SQL.* San Francisco, CA: Morgan Kaufmann.

Terenziani, P., & Snodgrass, R. T. (2004). Reconciling Point-Based and Interval-Based Semantics in Temporal Relational Databases: A Treatment of the Telic/Atelic Distinction. *IEEE Transactions on Knowledge and Data Engineering*, *16*(5), 540–551. doi:10.1109/TKDE.2004.1277816

Toman, D. (1996). Point vs. Interval-based Query Languages for Temporal Databases. In R. Hull (Ed.), *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (pp. 58–67). New York, NY: ACM Press.

Wang, F., Zaniolo, C., & Zhou, X. (2005). Temporal XML? SQL Strikes Back! In *International Symposium on Temporal Representation and Reasoning* (pp. 23–25). Los Alamitos, CA: IEEE Computer Society Press.

ADDITIONAL READING

Bettini, C., Jajodia, S., & Wang, X. S. (2000). *Time Granularities in Databases, Data Mining, and Temporal Reasoning*. Berlin, Germany: Springer. doi:10.1007/978-3-662-04228-1

Böhlen, M. H., Gamper, J., & Jensen, C. S. (in press). Temporal databases. In J. Hammer, & M. Schneider (Eds.), *Handbook of Database Technology*. London, U.K.: Chapman and Hall.

Böhlen, M. H., & Jensen, C. S. (2003). Temporal data model and query language concepts. In H. Bidgoli (Ed.), *Encyclopedia of Information Systems*. New York, NY: Academic Press. doi:10.1016/B0-12-227240-4/00184-2

Chakravarthy, S., & Kim, S.-K. (1994). Resolution of Time Concepts in Temporal Databases. *Information Sciences*, *80*(1-2), 91–125. doi:10.1016/0020-0255(94)90059-0

Chomicki, J., & Toman, D. (2005). Temporal databases. In M. Fisher, D. Gabbay, & L. Vila (Eds.), *Handbook of Time in Artificial Intelligence*. Amsterdam, The Netherlands: Elsevier.

Clifford, J., Dyreson, C. E., Isakowitz, T., Jensen, C. S., & Snodgrass, R. T. (1997). On the Semantics of "Now" in Databases. *ACM Transactions on Database Systems*, *22*(2), 171–214. doi:10.1145/249978.249980

Clifford, J., & Tuzhilin, A. (Eds.). (1995). *Recent Advances in Temporal Databases*. Berlin, Germany: Springer. doi:10.1007/978-1-4471-3033-8

Etzion, D., Jajodia, S., & Sripada, S. (Eds.). (1998). *Temporal Databases – Research and Practice*. Berlin, Germany: Springer. doi:10.1007/BFb0053695

Golfarelli, M., & Rizzi, S. (2009). A Survey on Temporal Data Warehousing. *International Journal of Data Warehousing and Mining*, *5*(1), 1–17. doi:10.4018/jdwm.2009010101

Gregersen, H., & Jensen, C. S. (1999). Temporal Entity-Relationship Models—A Survey. *IEEE Transactions on Knowledge and Data Engineering*, *11*(3), 36–44. doi:10.1109/69.774104

Jensen, C. S., & Snodgrass, R. T. (1999). Temporal data management. *IEEE Transactions on Knowledge and Data Engineering*, *11*(1), 36–44. doi:10.1109/69.755613

Jensen, C. S., & Snodgrass, R. T. (2009). Temporal Database. In L. Liu, & M. T. Özsu (Eds.), *Encyclopedia of Database Systems*. Berlin, Germany: Springer.

Johnston, T., & Weis, R. (2010). *Managing Time in Relational Databases*. San Francisco, CA: Morgan Kaufmann.

Özsoyoğlu, G., & Snodgrass, R. T. (1995). Temporal and real-time databases: A survey. *IEEE Transactions on Knowledge and Data Engineering*, *7*(4), 513–532. doi:10.1109/69.404027

Pissinou, N., Snodgrass, R. T., Elmasri, R., & Mumick, I. S. Özsu, M. T., Pernici, B., Segev, A., & Theodoulidis, B. (1994). Towards an Infrastructure for Temporal Databases: Report of an Invitational ARPA/NSF Workshop. *ACM SIGMOD Record, 23*(1), 35–51.

Roddick, J. F., & Patrick, J. D. (1992). Temporal Semantics in Information Systems—A Survey. *Information Systems*, *17*(3), 249–267. doi:10.1016/0306-4379(92)90016-G

Rolland, C., Bodart, F., & Léonard, M. (Eds.). (1987). *IFIP TC 8/WG 8.1 Working Conference on Temporal Aspects in Information Systems*. Amsterdam, The Netherlands: North-Holland.

Segev, A., Jensen, C. S., & Snodgrass, R. T. (1995). Report on The 1995 International Workshop on Temporal Databases. *SIGMOD Record*, *24*(4), 46–52. doi:10.1145/219713.219754

Snodgrass, R. T. (1990). Temporal databases: Status and research directions. *SIGMOD Record*, *19*(4), 83–89. doi:10.1145/122058.122068

Snodgrass, R. T. (1992). Temporal databases. In A.U. Frank, I. Campari & U. Formentini. *International Conference on GIS: From Space to Territory* (pp. 22–61). Berlin, Germany: Springer.

Snodgrass, R. T., & Ahn, I. (1986). Temporal Databases. *IEEE Computer*, *19*(9), 35–42. doi:10.1109/MC.1986.1663327

Tang, Y., Tang, N., & Ye, X. (2011). *Temporal Information Processing Technology and Its Applications*. Berlin, Germany: Springer.

Tansel, A., Clifford, J., Gadia, S. K., Jajodia, S., Segev, A., & Snodgrass, R. T. (Eds.). (1993). *Temporal databases: Theory, design, and implementation*. Redwood City, CA: Benjamin/Cummings.

Zaniolo, C., Ceri, S., Faloutsos, C., Snodgrass, R. T., Subrahmanian, V. S., & Zicari, R. (1997). *Advanced Database Systems*. San Francisco, CA: Morgan Kaufmann.

KEY TERMS AND DEFINITIONS

Bitemporal Table: A relational table with system support of one valid- and one transaction-time dimension.

Temporal Database: A database with built-in support for managing time-varying data.

Temporal Data Model: A data model for the representation of time-varying data.

Temporal Query Language: A query language that allows manipulation of time-referenced data.

Transaction Time: Temporal dimension concerning when some data is current in the database.

Transaction Time Table: A relational table with system support of one transaction time dimension.

TSQL2: A temporal extension of the SQL standard designed by a committee of temporal database experts chaired by R.T. Snodgrass in 1995.

Valid Time: Temporal dimension concerning when some fact is true in the modeled reality.

Valid Time Table: A relational table with system support of one valid time dimension.