Temporal Database System Implementations

Michael H. Böhlen
Department of Mathematics and Computer Science
Aalborg University, Fredrik Bajers Vej 7E
DK-9220 Aalborg Ø, DENMARK
boehlen@iesd.auc.dk

Although research on temporal database systems has been active for about 20 years, implementations have not appeared until recently. This is one reason why current commercial database systems provide only limited temporal functionality. This paper summarizes extant state of the art of temporal database implementations. Rather than being very specific about each system we have attempted to provide an indication of the functionality together with pointers to additional information. It is hoped that this leads to more efforts pushing the implementation of temporal database systems in the near future.

In order to include as many prototypes as possible a broad definition for temporal database systems was adopted. As a consequence this summary not only includes descriptions of systems that qualify as temporal database systems in the first place but also descriptions of systems that are related to temporal database systems, e.g., a temporal database generator. You will find descriptions of the following implementations.

- ARCADIA
- Calanda
- ChronoLog
- HDBMS
- TDBMS
- TempCASE
- TempIS
- TimeDB
- TimeIT
- TimeMultiCal
- T-REQUIEM
- T-squared DBMS
- VT-SQL

The information about specific temporal database implementations was collected incrementally by first sending an email to all participants of the International Workshop on Temporal Databases, Zürich, September 1995 with a request for pointers to people/groups not included in the original mailing list. This approach should ensure a good coverage of research on temporal database implementations. Still it might be possible that we have missed relevant information. My sincere apologies for any omissions. Developers of temporal database prototypes are encouraged to contact me, so that their system can be included into this document. Special thanks to those people who provided the system-specific information, namely

Aziz Ait-Braham (a_ait-braham@mac.co.umist.ac.uk)
Alex Buchmann (buchmann@dvs1.informatik.th-darmstadt.de)
Carlo Combi (combi@elet.polimi.it)
Chris J. Robertson (cjr@sapphire.otago.ac.nz),
Nandlal L. Sarda (nls@cse.iitb.ernet.in)
Duri Schmidt (schmidt@ubilab.ubs.ch)
Richard T. Snodgrass (rts@cs.arizona.edu)
Andreas Steiner (steiner@inf.ethz.ch)
Abdullah Uz Tansel (uztbb@cunyvm.cuny.edu)
Costas Vassilakis (costas@di.uoa.gr)

Besides general descriptions the survey classifies each system according to some well-known selection criteria. An analysis of these criteria reveals interesting properties (respectively "non-properties") of actual implementations. For example most systems have not been evaluated against large databases. Also traditional database features like persistence, transactions, and concurrency are not always provided. This leaves us with the question what a system has to provide at least to be rated as a temporal database system. Another interesting point is that the set of query operations that has been investigated is highly unbalanced. This probably means that many query languages only provide limited functionality. The following overview summarizes the main properties of evaluated temporal database system implementations.

Operation
- temp selection: 11
- temp projection: 9
- temp join: 8
- temp union: 8
- temp negation: 3

Database Size
- $<$ 1 MBytes: 6
- 4 MBytes: 1
- 5 MBytes: 1
- 10 MBytes: 1
- $>$ 50 MBytes: 2

Statement Class
- temp queries: 13
- temp updates: 8
- temp views/rules: 5
- temp integ. const.: 2

DB Technology
- relational: 10
- object-oriented: 2
- ER model: 1

Timestamp Format
- points: 11
- periods: 12
- temp elements: 2

Time Dimension
- user-defined time: 11
- valid-time: 12
- transaction time: 6

Time Properties
- multiple calendars: 2
- diff. granularities: 5
- indeterminacy: 2
- interpolation: 2

Timestamped Unit
- tuple: 9
- attribute: 1
- event: 1

Availability
- ftp available: 6
- available on request: 1
- not available: 6

Based on this overview it is possible to draw some conclusions about the status of current implementations and about future research directions. While not binding in any sense it might be worthwhile to take the following statements into consideration for future research activities on temporal database systems.

- The relational model with period-timestamped tuples is predominant.
- Transaction time support is negligable.
- Lots of work on temporal selections and temporal joins. Almost nothing on temporal negation.
- The focus is on queries. Updates, rules, and integrity constraints are neglected.
- Extant applications seem to indicate that, in general, different granularities are more important, or perhaps easier to support than, than multiple calendars, indeterminacy, and interpolation.
- The WWW, an increasingly important media, is unsupported.
- Temporal database design has barely found its way into products.

1 ARCADIA - GCH-OSQL

Institution Department of Biomedical Engineering - Politecnico di Milano

Developers F. Pinciroli, C. Combi , G. Pozzi

Contact address C. Combi (combi@elet.polimi.it), Ph.D., Dept. of Biomedical Engineering, Politecnico di Milano, P.zza L. da Vinci 32, 20133 Milano, Italy.

Description We implemented an object oriented temporal database at the top of ONTOS OODB. We also defined, developed and implemented an OO temporal query language named GCH-OSQL (Granular Clinical History - Object SQL). We faced the problem of managing temporal clinical information given at different and mixed granularity. In order to manage different and mixed granularities we defined a hierarchy of temporal classes, having at top the class interval. We used a three-valued logic to manage uncertainty coming from temporal relationships between data at different granularities. We applied our prototype to manage clinical data from PTCA patients (follow-up patients, after an intervention of coronary angioplasty).

Salient features

[ ] relational	[ ] multiple calendars	[ ] temp selection
[x] object-oriented	[x] different granularities	[ ] temp union
[x] time points	[x] indeterminacy	[ ] temp projection
[x] periods	[ ] interpolation	[ ] temp join
[ ] temp elements	[x] persistence	[ ] temp negation
[x] user-defined time	[ ] transactions	[x] temp queries
[x] valid-time	[ ] concurrency control	[ ] temp updates
[ ] transaction time	[ ] object versioning	[ ] temp integ. const.
[ ] tuple timestamps	[ ] schema modifications	[x] temp methods
[ ] attribute timestamps	[ ] views/rules	

other:

[x] object timestamping

Application domain

[ ] office automation	[ ] GIS	[x] other: temporal clinical data (medical records)
[ ] time-series	[ ] CAD	

DB size (bytes of data) you have worked with Few KBytes of testing data (data are real data from real patient, but only few data have been inserted to test our software)

Status of implementation, plans for the future The implementation status is that of a not completely tested prototype. The prototype has also a window-based user interface. We are planning to extend the implementation with other time-oriented services (displaying of temporal clinical data, natural language queries and so on). We are also planning to translate all this system for Ode OODB.

Hardware requirements Sun workstations with SunOS 4.1.3

Software requirements ONTOS OODB and Glockenspiel C++ Compiler

Availability For the moment this software is not distributed in any form. We hope to make parts of this software available by internet (www.medinfopoli.polimi.it).

2 Calanda

Institution UBILAB (Union Bank of Switzerland Information Technology Laboratory)

Developers D. Schmidt, A. Kotz Dittrich, W. Dreyer, M. Bleichenbacher

Contact address D. Schmidt (schmidt@ubilab.ubs.ch)

Description Time series management system especially suited for financial applications.

Salient features

[ ] relational	[x] multiple calendars	[x] temp selection
[x] object-oriented	[x] different granularities	[ ] temp union
[x] time points	[ ] indeterminacy	[ ] temp projection
[ ] periods	[x] interpolation	[ ] temp join
[ ] temp elements	[x] persistence	[ ] temp negation
[ ] user-defined time	[x] transactions	[x] temp queries
[x] valid-time	[x] concurrency control	[ ] temp updates
[ ] transaction time	[ ] object versioning	[ ] temp integ. const.
[ ] tuple timestamps	[x] schema modifications	[ ] temp methods
[ ] attribute timestamps	[ ] views/rules	

other:

[x] event timestamping	[x] grouping of time series
[p] time scale conversion	[p] filter and convert time series
[p] user defined methods	[x] graphical user interface

([p]= planned)

Application domain

[ ] office automation	[ ] GIS	[ ] other:
[x] time-series	[ ] CAD	

DB size (bytes of data) you have worked with 200-300 MByte, more than 10000 time series.

Status of implementation, plans for the future Working prototype. Implementation will continue and planned features will be implemented.

Hardware requirements No information.

Software requirements No information.

Availability Not available.

3 ChronoLog

Institution Institut für Informationssysteme, ETH Zürich, 8092 Zürich

Developers Michael Böhlen

Contact address M. Böhlen (boehlen@iesd.auc.dk)

Description ChronoLog is an period-timestamped temporal deductive database system running as a frontend to the commercial database system Oracle. Temporal requests (schema modifications, queries, and updates) are compiled into (sequences) of SQL-commands that are executed by the database backend. Main features of ChronoLog are

- ChronoLog, a logic-based data manipulation language
- facts, rules, and integrity constraints
- valid time, user-defined time, and transaction time
- temporal join, negation, disjunction, selection, and projection
- period-based semantics
- an explicit coalescing operation
- a temporally complete query language
- temporal update operations
- ChronoSQL, a SQL–based temporal query language
- a textual and a graphical interface

Salient features

[x] relational	[ ] multiple calendars	[x] temp selection
[ ] object-oriented	[ ] different granularities	[x] temp union
[x] time points	[ ] indeterminacy	[x] temp projection
[x] periods	[ ] interpolation	[x] temp join
[ ] temp elements	[x] persistence	[x] temp negation
[x] user-defined time	[x] transactions	[x] temp queries
[x] valid-time	[x] concurrency control	[x] temp updates
[x] transaction time	[ ] object versioning	[x] temp integ. const.
[x] tuple timestamps	[x] schema modifications	[ ] temp methods
[ ] attribute timestamps	[x] views/rules	

Application domain

[ ] office automation	[ ] GIS	[x] other: relational database applications and knowledge bases
[ ] time-series	[ ] CAD	

DB size (bytes of data) you have worked with Usually small.

Status of implementation, plans for the future The implementation has been completed and released to the public February 1995. No major extensions of the system are planned.

Hardware requirements All hardware platforms running SWI-Prolog.

Software requirements You need SWI-Prolog to run the core version. Oracle and Tcl, Tk, and Tcl-DP (graphical interace) are optional.

Availability ChronoLog is available from http://www.iesd.auc.dk/~boehlen.

4 HDBMS

Institution Indian Institute of Technology , Bombay, India

Developers Informatics project, Computer Science and Engineering Department, I. I. T, Bombay.

Contact address Dr. N. L. Sarda (nls@cse.iitb.ernet.in), C omputer Science and Engineering Department, I. I. T, Powai, Bombay - 400 076.

Description HDBMS (Historical Database Management System) is an extension to the relational model for modeling time and history data. It provides timing information, automatically maintains history (i.e., past) data, and it supports time-related queries. HDBMS uses an extended relational data model with state-oriented conceptualization. Two types of historical relations, called state and event relations, are provided for modeling real-world objects. The popular query langauge SQL has been extended for definition, retrieval and update of historical relations. HSQL contains a few new clauses, many operations and built-in functions on time domain, and facilities for retrospective updates.

Salient features

[x] relational	[ ] multiple calendars	[x] temp selection
[ ] object-oriented	[x] different granularities	[x] temp union
[x] time points	[ ] indeterminacy	[x] temp projection
[x] periods	[ ] interpolation	[x] temp join
[ ] temp elements	[x] persistence	[ ] temp negation
[x] user-defined time	[x] transactions	[x] temp queries
[x] valid-time	[x] concurrency control	[x] temp updates
[x] transaction time	[ ] object versioning	[ ] temp integ. const.
[x] tuple timestamps	[x] schema modifications	[ ] temp methods
[ ] attribute timestamps	[ ] views/rules	

Application domain

[ ] office automation	[ ] GIS	[x] other: general purpose
[ ] time-series	[ ] CAD	

DB size (bytes of data) you have worked with System has been tested with 4 tables varying in size from 200 tuples to 20,000 tuples. We have used the BTrieve file software, and no problems are anticipated for larger tables.

Status of implementation, plans for the future HDBMS has been distributed to many educational institutes in India for student learning (Netware platform). It has been ported to Unix (Objectrieve is used for record management). Future extensions are based on Unix platforms. Currently a master's student project is under way for schema evolution in temporal databases. Some extensions will be implemented in the HDBMS system.

Hardware requirements For PC version: IBM PC-XT, PC-AT, or any 100% compatible microcomputer. At least 640 KB of RAM. A hard disk with about 6 to 7 MB of available disk space.

Software requirements For PC version: MS-DOS version 2.0 or higher. Btrieve Software

Availability Internet: ftp://meru.cse.iitb.ernet.in/pub_in/hdbms. For further information please send an email to vijaya@cse.iitb.ernet.in.

5 TDBMS

Institution Baruch College, CUNY; University of Alberta; Bilkent University

Developers Abdullah Uz Tansel, Iqbal Goralwalla

Contact address A. Tansel (`uztbb@cunyvm.cuny.edu`), 17 Lexington Avenue Box E-0435 NY NY 10010 and I. Goralwalla (`iqbal@cs.alberta.ca`), Department of Computer Science, 615 General Services Building, University of Alberta, Edmonton, Alberta, T6G 2H1. Canada.

Description A temporal database management system based on N1NF relations with one level of nesting and attribute timestamping. It supports basic temporal relational algebra operations as well as restructuring operations PACK and UNPACK. It also includes an operation, ENUMERATION, for temporal selection and aggregate calculations.

Salient features

[x] relational	[ ] multiple calendars	[x] temp selection
[ ] object-oriented	[ ] different granularities	[ ] temp union
[x] time points	[ ] indeterminacy	[x] temp projection
[x] periods	[x] interpolation	[x] temp join
[ ] temp elements	[x] persistence	[x] temp negation
[x] user-defined time	[x] transactions	[x] temp queries
[x] valid-time	[ ] concurrency control	[x] temp updates
[ ] transaction time	[ ] object versioning	[ ] temp integ. const.
[x] tuple timestamps	[x] schema modifications	[ ] temp methods
[x] attribute timestamps	[ ] views/rules	

Application domain

[ ] office automation	[ ] GIS	[x] other: Commercial data processing
[ ] time-series	[ ] CAD	

DB size (bytes of data) you have worked with Approximately 10MByte.

Status of implementation, plans for the future Temporal relational algebra operations are complete. Integrity constraints, temporal indexes and query optimization are planned for future implementation.

Hardware requirements Sun Sparc workstations

Software requirements Unix operating system and ERAM (G. Ozsoyoglu, M. Ozsoyoglu, Department of Computer Engineering, Case Western Reserve University Cleveland, Ohio)

Availability Please contact `uztbb@cunyvm.cuny.edu` or `iqbal@cs.alberta.ca`.

6 TempCASE

Institution Department of Computation, UMIST

Developers TimrLab

Contact address Dr. Babis Theodoulidis (`babis@sna.co.umist.ac.uk`), Department of Computation, UMIST, PO Box 88, Manchester M60 1QD, United Kingdom. tel: +44 161 200 3309/3335, fax: +44 161 200 3324

Description TempCASE consists of a data definition tool for specifying graphically and textually an application domain, a data manipulation tool for manipulating application knowledge and a database design tool for transforming conceptual specifications to relational valid-time specifications. Data definition and manipulation is based on a conceptual design formalism which consist of an Entity-Relationship Time (ERT) model, a Conceptual Rule Language (CRL), a Process Interaction Diagram (PID). In all three models a particular emphasis has been put to time modelling.

The logical design in TempCASE consists of an automatic schema generation program which takes and ERT schema as input and produces a relational schema as output. The relational schema is in 3NF. Tables in the relational schema are automatically allocated primary primary and alternative keys.

Salient features

[ ] relational	[ ] multiple calendars	[x] temp selection
[ ] object-oriented	[x] different granularities	[x] temp union
[x] time points	[ ] indeterminacy	[x] temp projection
[x] periods	[ ] interpolation	[x] temp join
[x] temp elements	[x] persistence	[ ] temp negation
[x] user-defined time	[ ] transactions	[x] temp queries
[x] valid-time	[ ] concurrency control	[x] temp updates
[ ] transaction time	[x] object versioning	[ ] temp integ. const.
[ ] tuple timestamps	[x] schema modifications	[ ] temp methods
[ ] attribute timestamps	[x] views/rules	

other:

[x] entity-relationship approach

Application domain

[x] office automation	[ ] GIS	[ ] other:
[ ] time-series	[ ] CAD	

DB size (bytes of data) you have worked with About 5 MBytes.

Status of implementation, plans for the future Most features of the system have been successfully implemented. However there is room for improvement in performance, query optimisation, and data presentation. Other envisaged extensions to the work include (1) making the query language even easier to use by developing visual interfaces for it, (2) making the workbench independent of any specific DBMS, and (3) empowering it with a reverse engineering tool.

Hardware requirements Sun

Software requirements UNIX SUNOS 4.1.3 / Solaris 2.4, optional: INGRES 6.4 / ORACLE 7

Availability Currently not available.

7 TempIS Temporal DBMS

Institution Department of Computer Science, University of Arizona

Developers R. T. Snodgrass, I. Ahn, R. Stam, M. Soo

Contact address R. Snodgrass (`rts@cs.arizona.edu`)

Description TempIS is an extension of University Ingres (Version 8) that supports major portions of the TQuel temporal query language.

Salient features

[x] relational	[ ] multiple calendars	[x] temp selection
[ ] object-oriented	[ ] different granularities	[x] temp union
[x] time points	[ ] indeterminacy	[x] temp projection
[x] periods	[ ] interpolation	[x] temp join
[ ] temp elements	[x] persistence	[ ] temp negation
[x] user-defined time	[x] transactions	[x] temp queries
[x] valid-time	[x] concurrency control	[x] temp updates
[x] transaction time	[ ] object versioning	[ ] temp integ. const.
[x] tuple timestamps	[x] schema modifications	[ ] temp methods
[ ] attribute timestamps	[ ] views/rules	

Application domain

[ ] office automation	[ ] GIS	[x] other: general temporal applications
[ ] time-series	[ ] CAD	

DB size (bytes of data) you have worked with Medium, up to 4MByte relations.

Status of implementations, plans for the future There are no plans to extend this system, or prepare future releases.

Hardware requirements The system runs on VAX hardware under VM/UNIX, fourth release.

Software requirements The software is written in C.

Availability The system is covered by the 4.3BSD license, which can be obtained from the University of California at Berkeley. The extensions for TQuel are in the public domain.

8 TimeDB

Institution ETH Zürich, University of Arizona, Aalborg University
A component of the TimeCenter Architecture

Developers A. Steiner (implementor), M. Böhlen, C. S. Jensen, R. T. Snodgrass

Contact address A. Steiner (`steiner@inf.ethz.ch`), Institut für Informationssysteme, 8092 Zürich, Switzerland

Description TimeDB supports the query language ATSQL2, which is the result of integrating three different approaches, namely

- TSQL2, a consensus temporal query language
- ChronoLog, introducing the concept of temporal completeness
- Bitemporal ChronoSQL, featuring a bitemporal query language

TimeDB runs as a frontend to the commercial database system Oracle. ATSQL2 statements (queries, updates, and assertions) are compiled into (sequences of) SQL-92 statements which are executed by the backend. This approach guarantees a high portability between different platforms and different DBMS backends. TimeDB not only provides bitemporal statements but it also excels in a seamless integration of time into databases by supporting *upwards compatibility* and *temporal upwards compatibility.*

Salient features

[x] relational	[p] multiple calendars	[x] temp selection
[ ] object-oriented	[p] different granularities	[x] temp union
[x] time points	[ ] indeterminacy	[x] temp projection
[x] periods	[ ] interpolation	[x] temp join
[ ] temp elements	[x] persistence	[x] temp negation
[x] user-defined time	[x] transactions	[x] temp queries
[x] valid-time	[x] concurrency control	[x] temp updates
[x] transaction time	[ ] object versioning	[x] temp integ. const.
[x] tuple timestamps	[x] schema modifications	[ ] temp methods
[ ] attribute timestamps	[x] views/rules	

Application domain

[ ] office automation	[ ] GIS	[x] other: relational database applications
[ ] time-series	[ ] CAD	

([p]= planned)

DB size (bytes of data) you have worked with A few KBytes of data.

Status of implementation, plans for the future All features described above are implemented. Implementation is continuing. In particular, we have planned to add a graphical user-interface and to support access through the world wide web.

Hardware requirements Platforms running SWI-Prolog (e.g., Sun, PC) or SICStus-Prolog (e.g., Sun, Mac, PC).

Software requirements SWI-Prolog (`ftp://swi.psy.uva.nl/pub/SWI-Prolog`).

Availability `ftp://ftp.cs.arizona.edu/timecenter/timedb-beta.tar.gz` (a pre-beta release). The software is free and in the public domain.

9 TimeIT

Institution Department of Computer Science, University of Arizona
A component of the TimeCenter Architecture

Developers N. Kline, M. Soo

Contact address N. Kline (`kline@cs.arizona.edu`) and
M. Soo (`soo@babbage.csee.usf.edu`)

Description TimeIT is a system for testing algorithms and applications in the context of a temporal database. The system contains a temporal database generator that is driven by a substantial set of parameters, such as the range and distribution of the timestamps and the explicit attributes. Another component of TimeIT is a tool to graph one or two attributes of the generated relation. Finally, a runtime environment that collects statistics on metrics of interest, such as the number of sequential and random IO's and the number of tuple movements and comparisons, permits the performance of temporal database algorithms to be measured. A temporal join algorithm is provided as an example application.

Salient features

[x] relational
[ ] object-oriented

[ ] time points
[x] periods
[ ] temp elements

[ ] user-defined time
[x] valid-time
[ ] transaction time

[x] tuple timestamps
[ ] attribute timestamps

[ ] multiple calendars
[ ] different granularities
[ ] indeterminacy
[ ] interpolation

[ ] persistence
[ ] transactions
[ ] concurrency control

[ ] object versioning
[ ] schema modifications
[ ] views/rules

[ ] temp selection
[ ] temp union
[ ] temp projection
[x] temp join
[ ] temp negation

[x] temp queries
[ ] temp updates
[ ] temp integ. const.
[ ] temp methods

Application domain

[ ] office automation
[x] time-series
[ ] GIS
[ ] CAD
[x] other: testing any temporal algorithms

DB size (bytes of data) you have worked with 64 MByte relations

Status of implementations, plans for the future The current release is 0.1, a pre-beta release. There is only sketchy documentation available. In the future, we plan for more applications and documentation to be available, as well as enhancements to the tools.

Hardware requirements TimeIT functions currently on Alpha OSF/1 (V3.0+) and Solaris 2.3+. It should easily port to any BSD or SV.4 based Unix.

Software requirements The system requires GCC, G++, and the GNU libg++. The C++ and libg++ are used for generating temporal statistical distributions. The current source is mostly C with a small amount of C++.

Availability `ftp://ftp.cs.arizona.edu/timecenter/timeit-0.1.tar.gz`.
TimeIT is public domain software.

10 TimeMultiCal

Institution Department of Computer Science, University of Arizona
A component of the TimeCenter Architecture

Developers R. T. Snodgrass, C. Dyreson, C. S. Jensen, N. Kline, M. Soo, J. Whelan

Contact address R. Snodgrass (`rts@cs.arizona.edu`)

Description TimeMultiCal is a novel approach to providing limited extensibility for supporting multiple calendars and internationalization of time constants as well as a query processor prototype that demonstrates this approach. Our approach is notable in that we allow many different calendars to be used in the database management system. Ten languages are supported, accounting for about 1.5 billion native speakers. To illustrate how an existing DBMS could be augmented to support multiple calendars, we provide a prototype DBMS that supports the proposed extensions. This prototype consists of query analysis and execution components. It eschews traditional functionality such as concurrency control and disk access methods, as these aspects are orthogonal to timestamp management.

Salient features

[x] relational
[ ] object-oriented

[x] time points
[x] periods
[ ] temp elements

[x] user-defined time
[ ] valid-time
[ ] transaction time

[ ] tuple timestamps
[ ] attribute timestamps

[x] multiple calendars
[x] different granularities
[x] indeterminacy
[ ] interpolation

[ ] persistence
[ ] transactions
[ ] concurrency control

[ ] object versioning
[x] schema modifications
[ ] views/rules

[x] temp selection
[ ] temp union
[ ] temp projection
[ ] temp join
[ ] temp negation

[x] temp queries
[ ] temp updates
[ ] temp integ. const.
[ ] temp methods

Application domain

[ ] office automation
[ ] time-series
[ ] GIS
[ ] CAD
[x] other: general temporal applications

DB size (bytes of data) you have worked with Small databases, to test functionality.

Status of implementations, plans for the future TimeMultiCal, currently in release 1.1, consists of about 48K source lines of C code; the query processor prototype consists of about 63K source lines of code. The documentation consists of fifteen documents, comprising some 300 pages of material. We plan to migrate the system to supporting TSQL2.

Hardware requirements TimeMultiCal runs on Sun-4 machines under SunOS and DEC Alpha machines under OSF/1 (V3.0+).

Software requirements To build TimeMultiCal, you will need a compiler that supports 64 bit integers, such as the GNU C compiler (gcc), version 2.0 or above.

Availability `ftp://ftp.cs.arizona.edu/timecenter/timemultical-1.1.tar.gz`.
TimeMultiCal is free software in the public domain.

11 T-REQUIEM

Institution TH Darmstadt

Developers Dieter Dengel

Contact address H. Branding (`branding@dvs1.informatik.th-darmstadt.de`) and D. Dengel (`dengel@dvs1.informatik.th-darmstadt.de`)

Description T-Requiem (Temporal RElational Query and Update Interactive systEM) has been implemented as part of a master thesis. Handling of temporal features has been built into Requiem, a relational DBS which is public domain. Requiem is a single user system. Requiem served as a platform for the master thesis for didactical reasons. The main goal was the investigation of query processing techniques. The master thesis is part of a project on time-cognizant databases.

Salient features

[x] relational	[ ] multiple calendars	[x] temp selection
[ ] object-oriented	[ ] different granularities	[x] temp union
[x] time points	[ ] indeterminacy	[x] temp projection
[x] periods	[ ] interpolation	[x] temp join
[x] temp elements	[x] persistence	[ ] temp negation
[x] user-defined time	[ ] transactions	[x] temp queries
[x] valid-time	[ ] concurrency control	[ ] temp updates
[x] transaction time	[ ] object versioning	[ ] temp integ. const.
[x] tuple timestamps	[x] schema modifications	[ ] temp methods
[ ] attribute timestamps	[x] views/rules	

Application domain

[ ] office automation	[ ] GIS	[x] other: network managment
[ ] time-series	[ ] CAD	

DB size (bytes of data) you have worked with 500 KByte.

Status of implementation, plans for the future The query language compiler of Requiem has been modified to accept SQL statements. Handling of a subset of TSQL2 has been added. The next step is to improve the implementation concepts and port it to a real-time DBS prototype (integration into transaction management). Of special interest is the handling of temporal integrity constraints.

Hardware requirements PC, workstations.

Software requirements The system has been developed on Solaris 2.4 with Sun C-Compiler. It runs with minor changes on any C-file system (UNIX, MS-DOS).

Availability For the moment not be made available because it is not in the final stage of implementation.

12 T-squared DBMS

Institution Database Research Group, Dept. of Computer Science, U. of Otago

Developers Dr. C. J. Robertson

Contact address Dr. C. J. Robertson (`cjr@sapphire.otago.ac.nz`), Database Research Group, Department of Computer Science, University of Otago P.O.Box 56, Dunedin, New Zealand

Description We have implemented a temporal relational database that supports six relation types: static, snapshot, valid-time, transaction-time, bitemporal and transformational. The transformational relation type allows the graceful, rule-based, degradation of data. There are four "levels" of interface: relational algebra, SQL, temporal SQL, and a transformationally extended temporal SQL.

Salient features

[x] relational	[ ] multiple calendars	[x] temp selection
[ ] object-oriented	[ ] different granularities	[x] temp union
[ ] time points	[ ] indeterminacy	[x] temp projection
[x] periods	[ ] interpolation	[x] temp join
[ ] temp elements	[x] persistence	[ ] temp negation
[x] user-defined time	[ ] transactions	[x] temp queries
[x] valid-time	[ ] concurrency control	[x] temp updates
[x] transaction time	[ ] object versioning	[ ] temp integ. const.
[x] tuple timestamps	[ ] schema modifications	[ ] temp methods
[ ] attribute timestamps	[x] views/rules	

Application domain

[ ] office automation	[ ] GIS	[x] other: general purpose
[ ] time-series	[ ] CAD	

DB size (bytes of data) you have worked with Usually small.

Status of implementation, plans for the future Although the DDL and DML are fully implemented, the state of this system is considered to be that of a working prototype. In addition we have added a relational algebra interface to enable us to experiment with temporal relational operators.

We plan to upgrade the database to work with Oracle V.7 and extend its functionality.

Hardware requirements The database has been developed on a DECstation 3100, running ULTRIX V4.3 with 16Mb of Ram and a 600Mb hard disk.

Software requirements It is a loosely-coupled front-end written in C and uses dynamic SQL to communicate with an Oracle V6.0 relational database.

Availability Currently not available.

13 VT-SQL

Institution ORES Project (ESPRIT III - 7224)

Developers University of Athens, Agricultural University of Athens, Information Dynamics (INGRES Hellas)

Contact address Email: `ores@di.uoa.gr`

Description VT-SQL is an extension of SQL, allowing for the definition of tables containingvalid time data, as well as for the manipulation of these data. The VT-SQL application operates as a front-end shell to the INGRES relational DBMS. The shell supports the temporal senantics, while the INGRES kernel is used for data storage and retrieval. The INGRES kernel has been extended to support an additional data type, namely DATEINTERVAL, which is a time interval with a granularity of date.

Salient features

[x] relational	[ ] multiple calendars	[x] temp selection
[ ] object-oriented	[ ] different granularities	[x] temp union
[x] time points	[ ] indeterminacy	[x] temp projection
[x] periods	[ ] interpolation	[x] temp join
[ ] temp elements	[ ] persistence	[ ] temp negation
[x] user-defined time	[x] transactions	[x] temp queries
[x] valid-time	[x] concurrency control	[x] temp updates
[ ] transaction time	[ ] object versioning	[ ] temp integ. const.
[x] tuple timestamps	[ ] schema modifications	[ ] temp methods
[ ] attribute timestamps	[ ] views/rules	

Application domain

[ ] office automation	[ ] GIS	[x] other: Hospital application
[ ] time-series	[ ] CAD	

DB size (bytes of data) you have worked with Up to 100 KB/table.

Status of implementation, plans for the future The DDL and DML have been fully implemented (with no support for views, however), within the ORES project. Later, in the context of a PhD thesis, the algorithms for data manipulation have been optimised, transaction support has been built in, and algorithms for concurrent user support have been designed. Future work focuses in the full implementation of the concurrent users support, transaction time and porting of the system on top of an object-oriented DBMS.

Hardware requirements VT-SQL has been developped on a SUN Sparc machine, running SUNOS 4.1.3. It is known to run under the Solaris 2.x operating system as well, without any modifications. With minor ammendments, it can run under Ultrix 4.x.

Software requirements INGRES version 6.4, with the kernel extensions (object files, linkable to the kernel).

Availability Since VT-SQL was developped within an ESPRIT project, it falls under the rules imposed to ESPRIT products classified as 'restricted'. Further information is available from the project prime contractor, "01-Pliroforiki".

```
01-Pliroforiki                    Tel: +301 2182500
Aharnon 438                       Fax: +301 2533161
111-43 Athens, Greece
```