Querying Temporal Clinical Databases with Different Time Granularities: the GCH-OSQL Language

Carlo Combi (°), Francesco Pinciroli (°*), Marco Cavallaro, and Giorgio Cucchi

° Dipartimento di Bioingegneria del Politecnico di Milano
* Centro di Teoria dei Sistemi del CNR, Milano

There is a need for managing temporal clinical information given at different levels of granularity. Different time granularities are also needed in querying clinical databases. In this paper, we describe GCH-OSQL (Granular Clinical History - Object Structured Query Language), an object-oriented temporally-oriented extension of SQL. GCH-OSQL is based on an object-oriented temporal data model. It allows storage of clinical information at different and mixed granularities. GCH-OSQL deals with the valid time of clinical information. GCH-OSQL offers also a graphical user-interface. It guides different end users, from expert to naive, to formulate expressive and correct queries.

INTRODUCTION

The field of medical informatics has addressed the management and representation of clinical information temporal dimension [1, 2]. Time influences decision making activity in diagnosis, therapy, and prognosis [2, 3]. Recently, many researchers highlighted the need for querying temporal clinical databases by high abstraction level languages [2, 3].

Clinical information consists both of natural language sentences (such as anamneses, therapies), and of quantitative parameters (such as heart rate, systolic blood pressure). The temporal dimension of such data could be expressed by using different time units, or by expressing temporal localization with some degree of uncertainty: different levels of *granularity* characterize the clinical information [4]. The granularity of a given temporal information is the level of abstraction at which information is defined [4]. For database systems based on the calendar time, granularity is the accuracy or the unit of measure used for the time scale (Ex: days, hours, minutes, ...).

Some attempts have been made to manage and store temporal clinical information with different granularity at the database level [2, 4]. A specific need consists in the capability of querying temporal clinical databases specifying suitable time granularities, not directly related to the different granularities of the stored clinical information (Ex: "select the patients, who three years ago ...", but "select the patients, who within three hours ...").

An object-oriented approach seems to be appropriate in managing the complexity of temporal clinical information [3, 4]. For this type of approach, we defined and implemented the object-oriented and temporally-oriented query language GCH - OSQL (Granular Clinical History - Object Structured Query Language) as a temporal object-oriented extension of the well-known query language SQL. The main aims of the GCH-OSQL are:

- to allow the user to make queries on temporal object-oriented clinical databases, involving different and mixed temporal granularities;
- to manage at the database level the uncertainty coming from comparisons between temporal data of different granularities.

We also developed a graphical interface for the GCH-OSQL, to guide users with different skills easily conduct correct temporal queries.

THE OBJECT-ORIENTED TEMPORAL DATA MODEL

Basic Concepts

The object-oriented temporal data model of GCH-OSQL is based on a common, usual object-oriented data model as in [5]. An *object* models any kind of entity, e.g. a patient, a pathology. Each object has a unique *identity*, provided by the OID (Object IDentifier). A class (or a type) describes objects, sharing the same properties: for example, the class *patient* describes the common features of all the patients, represented into the database. Objects are then instances of a class. Properties of an object and relationships with other objects are defined by the public interface of the class, consisting of *methods*. Our model supports the most common features of object-orientation: information hiding, overloading, encapsulation, and single inheritance [5]. In the following we will use the dot notation in referring to the properties of an object by a method: the notation *o.m*(..,..) stands for the application of the method *m*, with the suitable arguments, to the object *o*. For example, the birthdate of a certain patient *p* will be given by the notation *p.birth_date*().

We extended the model to manage the temporal dimension of information given at different and mixed granularities. We distinguish *temporal objects* and *atemporal objects*. By atemporal objects we model the real-world entities without a considered temporal

0195-4210/95/$5.00 © 1995 AMIA, Inc.

dimension; by temporal objects we model real-world entities, for which the temporal dimension has to be considered. The model supports the *valid time* for temporal objects [6]: the valid time of an object is the time when the fact, modeled by the object, is true in the represented reality. For example, the valid time of an object *symptom* is the time, during which the particular symptom is present.

Some predefined data types model the temporal dimension of information; particularly the type *interval* allows to model the valid time of a temporal object: each temporal object *o* has the particular method *o.valid_interval()* returning the valid time of the object one. The type *interval* is at the top of a hierarchy of types modeling the temporal dimension: the hierarchy is composed by the types *elementary instant*, *instant*, *duration*, *interval*. A further predefined type, *t_o_set*, allows for management of temporal object sets.

Finally, the suitable predefined type *bool3* manages uncertainty coming from relationships between temporal objects, using a three-valued logic.

Time modeling by predefined types

An interval can be specified by the starting instant, the ending instant, and the duration, given at different and mixed granularities. Some constraints exist among the starting and ending instants and the duration: given two of the three entities used to specify an interval (e.g. the starting instant and the duration), the third entity (e.g. the ending instant) is affected by the granularity of both the first two entities: using these three entities, we can express an interval with different and heterogeneous granularity.

It is possible to identify an interval having a duration given at a granularity finer than that of the starting and ending instants: for example, "on September 1994, for 24 minutes". On the other hand, it is possible to identify some intervals having the starting (ending) instant given at a granularity finer than the ending (starting) instant: "on September 23, 1994 for seven months" or "for seven months until September 23, 1994" or "from September 23, 1994 to December 12, 1994 in the evening".

Instants, specified at different granularity, are represented on the time axis by two time points - called elementary instants - expressed at the finest time unit of seconds. The instant "10/10/94" can coincide with one of the elementary instants between the two extremes "00:00:00 - 10/10/94" and "23:59:59 - 10/10/94". As for instants, two distances between elementary instants identify a duration.

A suitable class corresponds to each of the above described concepts, as previously mentioned.

The interval class. The class interval has many methods that allow to verify the existence of temporal relationships among intervals. In respect to the relationships between intervals introduced by Allen in [7], we defined a richer set of temporal relationships. This allows us to also consider relationships dealing with different granularities and with the duration of intervals (tab I).

Relationships
a temporally expressed as b
a contemporary to b at granularity X
a temporally specifies b
a before b
a overlaps b
a during b
a starts as b at granularity X
a finishes as b at granularity X
a meets b at granularity X
a has duration expressed as b
a lasts as b at granularity X
a specifies the duration of b
a lasts minus than b

Tab.I Relationships defined for the class interval. The table describes, by natural-language expressions, the relationships defined for the class interval. *a* and *b* are instances of the class interval; *X* is a given granularity.

The t_o_set class. The t_o_set class (temporal_object_set) allows for management of a (partially ordered) set of temporal objects. The common operations on sets apply to instances of the t_o_set class: insertion, deletion, intersection, union, difference, proof of the membership of an element to the set, proof that the set is empty, or contained in another set. Many other methods characterize the t_o_set class, allowing to verify the existence of temporal relationships among objects, having some specified features, belonging to an instance of the t_o_set class. An instance of the t_o_set class is a temporal object. Specialized temporal objects can be defined by inheriting from the t_o_set class. For example, it is possible to define the sets of the follow-up visits by a class inheriting from the t_o_set class, adding the constraint that each of these sets is completely ordered (it is not possible to have two overlapping follow-up visits).

The three-valued logic

Because of time stamping clinical data at different granularities, uncertainty can be related to relationships between intervals or instants. In fact, it is not always possible to be certain about truth or falsehood of relationships between intervals or instants. Consider the two sentences "In December 1994, the patient suffered from a headache for seven days", and "In December 1994, the patient suffered from flu for fifteen days". If we can say that flu affected the patient longer than headache did, we cannot answer TRUE or FALSE to the question of

whether the patient suffered from a headache before suffering from flu. Both answers could be wrong, because we don't have enough information (we don't know which day of December the two pathologies started or which day they ended). Therefore, we used a three-valued logic introducing, apart from the values TRUE and FALSE, the value UNDEFINED. In the object-oriented temporal data model, the predefined type *bool3* allows for management of all the three truth values, by extending the usual logical connectives (AND, NOT, OR) and the logical quantifiers (EXISTS, FOR EACH).

THE QUERY LANGUAGE GCH-OSQL

We extended the SQL query language only in the SELECT operation; The particular object-oriented DBMS will manage the other operations (insertion, deletion, schema updating, and so on) by a specific object-oriented programming language. Usually the end user manages the clinical database by suitable graphical interfaces [4].

The temporal extension of the SELECT construct includes the addition of the TIME-SLICE and MOVING WINDOW clauses, and the capability of referencing the temporal dimension of objects in the WHERE and SELECT clauses.

Queries in GCH-OSQL

The format of GCH-OSQL query presents the following clauses, where the square brackets mean that the clause is optional, in the specified order:

SELECT.....
FROM.....
[WHERE.....]
[TIME-SLICE.....]
[MOVING WINDOW.....]

The query returns the required information by the methods specified in the SELECT clause, from the instances in the database of the classes specified in the FROM clause satisfying the conditions imposed by the optional WHERE, TIME-SLICE, MOVING WINDOW clauses. These clauses consider all the objects for which the specified condition is TRUE or UNDEFINED. Some suitable logical connectives or keywords in the clauses allow the user to consider only the TRUE value or only the UNDEFINED value.

The WHERE clause. This clause can contain both the usual *atemporal conditions* and complex *temporal conditions*. Temporal conditions are based on the methods translating the previously mentioned relationships among temporal objects; in other words, temporal conditions (Ex: the pathology "myocardial infarction" occurred before the pathology "angina") can be expressed by using methods of the class interval applied to the valid time. More global temporal conditions can be expressed by the methods of the t_o_set class, verifying some features of the temporal clinical objects belonging to a certain set. For example, it is possible to select the patients who have, between their pathologies, both myocardial infarction and angina in that temporal order.

Complex conditions could be composed beginning from simpler conditions, by the use of the logical connectives AND, OR, NOT, MUSTBE, MAYBE, MUST_NOTBE. By the connectives MUSTBE, MAYBE, and MUST_NOTBE, it is possible to explicitly manage each of the three truth values.

The TIME-SLICE clause. With the TIME-SLICE clause, only those objects in the database whose valid time is contained in the interval specified in the clause, are considered. For example, it is possible to consider the patients' clinical facts for only the last two years.

In this clause is also possible to use the MUSTBE and MAYBE keywords. Using the MUSTBE keyword, only those objects whose valid time is contained in the interval specified in the clause come advised. Using the keyword MAYBE only those objects, for which it is uncertain that the valid time is contained in the specified interval, are selected.

The interval can be specified in four different ways: the user could specify an interval, at the desired granularity, by the keywords FROM and TO. The interval can also be specified only by the starting instant (FROM) or by the ending instant (TO). The AT keyword allows for a particular interval consisting of an instant specified at a suitable granularity.

The MOVING WINDOW clause. Through this clause the objects stored in the database are examined by a temporal window, of width specified in the clause, that moves along the time axis. The conditions specified in other clauses are verified only on the objects observable by this window. By this clause, for example, we can define the maximum delay between the considered clinical facts of a patient.

The temporal dimension of the interval specified in the MOVING WINDOW clause is specified by a number followed by a label related to the granularity. The labels for granularity, from the years to the seconds, are yy, mm, dd, hh, min, ss.

QUERYING A CLINICAL DATABASE

The object-oriented temporal data model previously defined has been applied to a clinical database. The clinical database prototype is able to manage clinical data for follow up of patients who have undergone coronary-artery angioplasty (PTCA) [4]. These patients, after a 2-3 day-long hospitalization, go through some periodical follow-up visits aimed at verifying the efficacy of angioplasty. For these patients, the anamnestic data have to be updated during the follow-up period. Management of temporal aspect of these data is therefore relevant; the clinical

history of these patients has to be managed to relate follow-up parameters with clinical events, such as pathologies or therapies [4]. The clinical database contains some temporal objects, modeled by the classes *therapy*, related to previous or current therapies, *diagnosis*, related to previous or current pathologies, *angio_visit*, related to the parameters - blood pressure, heart rate, .. - collected during follow-up visits. The database is patient-oriented: the class *patient* allows to access to all the data of the patients. Each object of the class *patient* can have many temporal objects of the classes *therapy*, *diagnosis*, *angio_visit*, partially ordered by suitable instances of the class t_o_set (or its subclasses).

To show the expressiveness of GCH-OSQL, let us consider the following clinical query. For a patient who has undergone PTCA, it is important that the blood pressure is normal (i.e. SBP: 100 - 150 mmHg, DBP: 60 - 100 mmHg) in the period following the intervention. We want to know which patients, having normal values of the blood pressure for 90 days following the intervention, suffered from angina and underwent a reintervention within three years. We will only consider the period between March 87 and May 95.

We can express the query by GCH-OSQL as:

```
SELECT P.surname(), P.name()
FROM patient P, diagnosis A, angio_visit B,
    angio_visit C
WHERE P.Dia_Set().OCCURS(A) AND
    P.Visit_Set().OCCURS(B) AND
    P.Visit_Set().OCCURS(C) AND
    B.angio_exam().exam_type() = "PTCA" AND
    C.angio_exam().exam_type() = "PTCA" AND
    A.pathology()="angina" AND MUSTBE
    (B.BEFORE(A) AND A.BEFORE(C)) AND
    P.Visit_set().MAINTAINS("SBP()>100 AND
    SBP()<150", B.start(),"90dd") AND
    P.Visit_set().MAINTAINS("DBP()>60 AND
    DBP()<100", B.start(),"90dd")
TIME SLICE FROM 3/87 TO 5/95
MOVING WINDOW 3yy
```

In this GCH-OSQL query we can observe some relevant features: i) different granularities (years, months, and days) are explicitly used; ii) the MUSTBE operator allows for verification of the certainty of the precedence between PTCA, angina, and the new PTCA; iii) for temporal relationships, the query uses both methods of the class interval (BEFORE is a method of the class interval - classes diagnosis and angio_visit inherit from this class) and of the class t_o_set (MAINTAINS and OCCURS are methods of the class t_o_set); iv) by the methods of the class t_o_set it is possible to verify some complex conditions on the objects belonging to a set (in this case that, during a period of 90 days starting from the PTCA intervention, the SBP and DBP values are within a predefined range); v) the condition by the MAINTAINS method considers also the patient for which it may be the blood pressure has been in a range of normality.

THE SYSTEM

We implemented GCH-OSQL on SUN workstations, using ONTOS [8], an object oriented DBMS, and OPEN LOOK, a set of tools for graphic applications development.

GCH-OSQL provides users with a set of different tools, related to the user skill, guiding composition of reliable queries. In the construction of the graphic interface of GCH-OSQL we tried to satisfy two specific skill requirements of users. An Elementary Guide is provided for a user already experienced in the system, requiring simple help in composing the clauses. A Complete Guide (Fig. 1) is provided for an unexperienced user, who wants to be driven in the formulation of queries.

DISCUSSION AND CONCLUSIONS

GCH-OSQL and the related temporal object-oriented data model are not able to represent at the data level temporal information having high abstraction levels (Ex: LOW, DECREASING, and FAST abstractions for the Hb-count parameter [9]). However, GCH-OSQL, applied to the temporal data querying at database level, could enable higher abstraction level systems to perform more powerful abstractions on the clinical data, also considering mixed granularities and uncertainty in the temporal references.

Our temporal object-oriented temporal data model is the only such model with the capability to identify, at the database level, a time interval by different granularities both for the starting instant, the duration, and the ending instant.

Regarding the granularity, our approach is similar to those in [2] in allowing for expression of the uncertainty in the time reference that is not only limited to the calendar-date granularity.

Some relevant features characterize GCH-OSQL, in respect with other contributions in medical informatics literature.

- **Management of valid time expressed with different granularity.** The valid time could be expressed at different granularity both during data insertion and query composition. A subsequent result is the simple management of temporal conditions in queries. In respect to SQL, in GCH-OSQL few additional clauses are present: TIME-SLICE and MOVING WINDOW. Instead adding other clauses, we have thought of express the verification of the temporal relationships inside the WHERE clause.

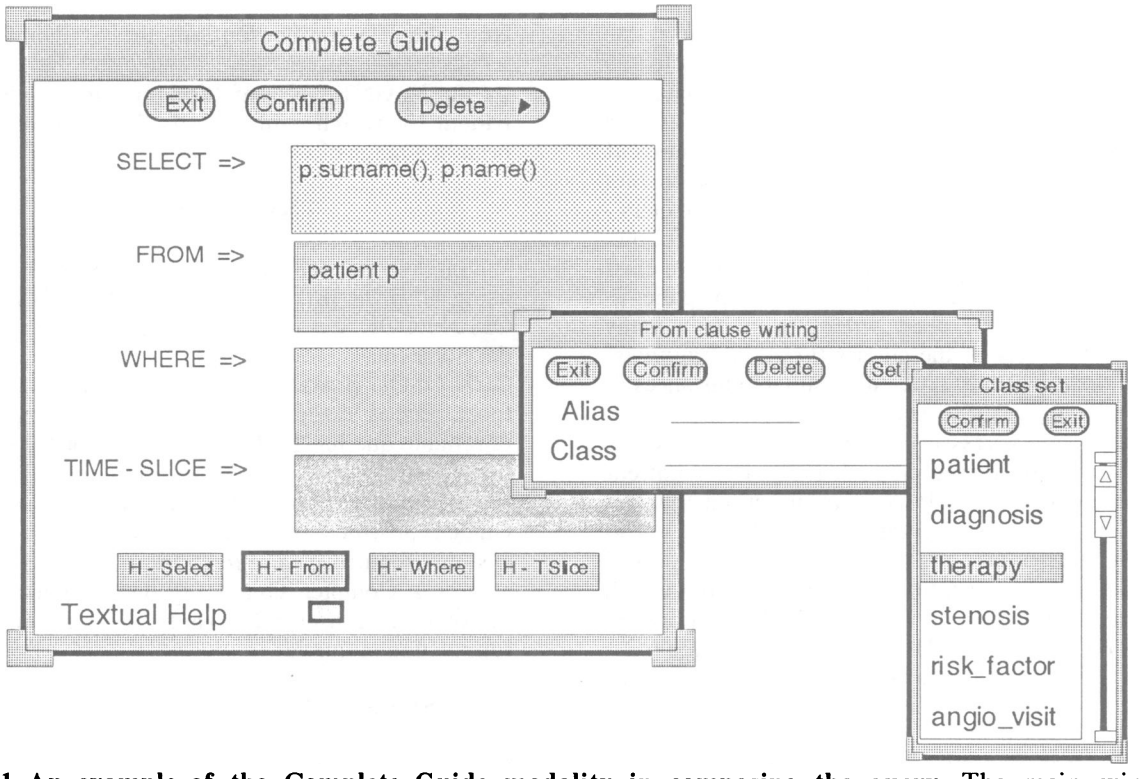

Fig. 1 An example of the Complete_Guide modality in composing the query. The main window globally displays the query; some other windows help to select a class name, during the composition of the FROM clause. The MOVING WINDOW clause isn't present in the prototype version of the system.

- **Management and expression of the uncertainty related to temporal relationships.** GCH-OSQL manages temporal relationship uncertainty, using a three-valued logic. The connectives MUSTBE, MAYBE, MUST_NOTBE explicitly manage the values of such logic. By this logic, it is possible to distinguish sure situations by possible situations, due to comparison between temporal dimensions expressed in different granularity.

Lastly, we provided a set of graphical interfaces, depending on the user skills, to support the composition of the query.

Acknowledgments

This work was partially supported by contributions from: MURST 40% Italian National Project for Medical Informatics; MURST 40% Italian National Project for the Bio-Engineering of Cardiovascular System; Department of Biomedical Engineering of the Politecnico di Milano; CNR's Centro Studi per la Teoria dei Sistemi.

References

1. Das AK, Shahar Y, Tu SW, Musen MA. A Temporal-Abstraction Mediator for Protocol-Based Decision-Support System. In: Ozbolt, JG (ed.), 18. Symposium on Computer Applications in Medical Care. Philadelphia, Hanley & Belfus, 1994; 320-324.
2. Das AK, Musen MA. A Temporal Query System for Protocol-Directed Decision Support. Methods of Information in Medicine. 1994; 33: 358-70.
3. Kahn MG, Tu SW, Fagan LM. TQuery: A Context-Sensitive Temporal Query Language. Computers and Biomedical Research. 1991; 24: 401-419.
4. Combi C, Pinciroli F, Musazzi G, Ponti C. Managing and displaying different time granularities of clinical information. In: Ozbolt, JG (ed.), 18. Symposium on Computer Applications in Medical Care. Philadelphia, Hanley & Belfus, 1994; 954 - 958
5. Kim W. Modern Database Systems. New York: Addison Wesley, 1995.
6. Jensen CS, Clifford J, Elmasri R, et al. A Consensus Glossary of temporal Databases Concepts. SIGMOD RECORD. 1994; 23: 52-64.
7. Allen J. Maintaining Knowledge about Temporal Intervals. Commun. ACM. 1983; 26: 832-843.
8. ONTOS Developer's guide, Ontologic ONTOS Database, Version 2.0, 1991.
9. Shahar Y, Musen MA. RESUME: A Temporal-Abstraction System for Patient Monitoring. Computers and Biomedical Research. 1993; 26: 255-273.