

# Cooperative Interfaces for Spatio-Temporal Databases

José Pérez-Alcázar<sup>2</sup>, Andrea S. Hemerly<sup>1,2</sup>, Marco A. Casanova<sup>1</sup>  
and Antonio L. Furtado<sup>2</sup>

jose@inf.puc-rio.br - andrea@vnet.ibm.com - casanova@vnet.ibm.com  
furtado@inf.puc-rio.br

<sup>1</sup>Centro Científico Rio  
IBM Brasil  
Av. Presidente Vargas, 824/844  
20071 Rio de Janeiro RJ - Brasil

<sup>2</sup>Departamento de Informática  
Pontifícia Universidade Católica do RJ  
R. Marquês de São Vicente, 225  
22453 Rio de Janeiro RJ - Brasil

## Abstract

This work addresses cooperative interfaces for spatio-temporal databases where cooperativeness means to help users achieve their goals. Starting from a rule-based cooperative interface for conventional databases, its extension to cope with spatio-temporal data is described. We concentrate on the novel opportunities for cooperativeness that the spatial and time dimensions provide.

## Keywords

Cooperative interfaces; expert database systems; query processing; Prolog. 1. Introduction

The manipulation of data-based information systems tends to become increasingly difficult to users, as a consequence of the addition of complex but often badly needed features such as the time and spatial dimensions. For example, the costs of implementing a geographic information system (GIS), where the spatio-temporal dimensions are important, follow approximately the proportion of 1:hardware, 10:software, 1,000,000:database, 10:training (Frank et al, 1992). The construction and maintenance of the database is the most expensive item. Hence, the development of tools that facilitate the use of geographic information systems is a relevant research topic. Moreover, the spatio-temporal semantic aspects suggest novel opportunities for cooperative interfaces, as will be shown in this paper.

An interface is cooperative (Webber, 1986;Cuppens and Demolombe, 1989) to the extent that it interacts with users in ways that: (1) contribute to the achievement of users' goals; (2) keep the users' understanding of the system in harmony with the definition and contents of the system, avoiding misconceptions and contributing to a

fuller and at the same time more efficient usage; (3) conform to the established integrity and authorization constraints.

There are several proposals for cooperative interfaces (Cuppens and Demolombe, 1989; Chu et al, 1990; Hemerly et al, 1991a). Here, we shall focus on the system proposed in (Casanova and Furtado, 1990; Hemerly et al, 1991b). The system is driven by a set of rules that transform requests (queries or updates) and responses; for simplicity, we shall limit our discussion to queries and answers (cooperative query answering systems). The rules may be specific to the application in question, they may be tuned to a given class of users, or they may be domain-independent. On the other hand, the rules may correct or complement a query, give alternatives to a failed query, provide explanations, solve ambiguities or edit an answer.

In order to show the cooperative techniques introduced in this article, we formulate examples in an SQL-like query language, with temporal and spatial operators. However, we believe that the cooperative techniques can be easily adapted to other TDMS languages. This language, STSQL, is based on TSQL (Temporal SQL) (Navathe and Ahmed, 1989), a SQL-like query language which is used to consult rollback relational databases - manipulating only valid time - and on a Spatial SQL (Egenhofer, 1989) which comes equipped with spatial operators, and is built out of the primitive types 'point', 'line' and 'polygon'. In what follows, we will also refer to these spatial attributes as the *geometries* of a spatial object.

æ

## 2. Cooperative behaviour through query modification

The approach adopted here, involves an algorithm, driven by rules of different kinds (to be indicated below), to perform query modification or edit the resulting answers. A prototype PROLOG/SQL implementation following this approach is operational. The specification of a cooperative interface system for a database application then consists of a set of *domain-independent rules*, which are proper to the environment, and a *database application context* containing: (1) the database conceptual schema, the external schemas and the application integrity constraints, expressed in the data model adopted; (2) an application domain model, which captures conceptual information not expressed in the conceptual schema, and which may include a set of *application-dependent rules*, relevant to all users of the application; and (3) one or more user models, which likewise capture information not expressed in the external schemas, and which may include a set of *user-dependent rules*, applicable only to specific classes of users.

There are two general classes of rules: the *request modification rules (RM-rules)*, which indicate the transformations queries may suffer, and the *template rules*, which define how to present query results. Orthogonally, we classify the RM-rules according to the execution phase where they are used. Thus, we have *pre-rules*, which are invoked before the execution of a query, *s-post-rules*, which follow the successful execution of a request, and *f-post-rules*, which follow execution in case of failure.

Among other functions, a cooperative environment may then: (1) **Correct** a query or simply translate a user-friendly external format into the standard syntax, usually employing pre-rules. (2) **Complement** a query by supplying additional information, through the use of pre-rules or, alternatively, through s-post-rules that generate new requests. (3) **Generate alternatives** to accomplish, as much as possible, the same purposes as the original query, when it fails, with the help of f-post-rules. (4) **Give explanations** about an answer or about the behavior of the system, employing s-post- or f-post-rules. (5) **Solve ambiguities** of a query or reveal the semantics of an answer, using pre- or s-post-rules. (6) **Edit** an answer, making it more informative, perhaps by omitting uninteresting or non-authorized items, through the use of template rules.

The request modification algorithm is formally described in (Hemerly et al, 1991a). Briefly, the algorithm successively applies all possible pre-rules to the query, generating a new one. Then, it executes the modified query against the database. After a successful execution, it can perform new actions as the result of the cumulative processing of s-post-rules. In case of a failed execution, the algorithm may apply f-post-rules, one at time, generating alternative queries, until one succeeds. If all alternatives fail, it also cumulatively tries f-post-rules to compensate for the failure. Finally, the algorithm uses template rules to edit query results. æ

### 3. Cooperative Query Formulation in Spatio-Temporal Databases

To illustrate the use of the rule-based cooperative answering system, we assume a simple-minded geographic database, whose schema is described as follows:

```
city( name, geometry, Ts, Te );
population( name_city, quantity, Ts, Te );
school( name, street, number, geometry);
health_institution( name, street, number, type, geometry);
hospital( name, number_of_beds, Ts, Te );
```

The relation hospital represent one specialization of health institution, which means that it inherits its attributes. The attribute *type* in health-institution serves to classify it. The attributes city's geometry, quantity and number-of-beds are assumed to be dynamic.

A number of example queries are then formulated in STSQL over this simple database. They cover some of the roles that cooperativeness can play. Apart from the use of STSQL, the presentation will be as informal as possible; for a more formal discussion, particularly of RM-rules, the reader is referred to (Hemerly et al, 1991a).

One of the role of the cooperative interface is the automatic correction of ill-formulated queries. This role perhaps provides the most natural examples of cooperative query processing. For example, suppose that there is a polymorphic operator *intersect*, that accepts as parameters polygons and lines, but not points, and a polymorphic operator

in, that accepts as parameters polygons, lines and points. Then consider the query ‘Retrieve all roads that crossed city C in 1990-1992’:

```
select road.name
from road, city
where city.name = 'C' and
      road.geometry intersect city.geometry
time_syear [1990-1992]
```

However, suppose that the geometry of a city is a point, which implies that the formulation of the query is mistaken. The system can detect this problem and it may then correct the problem by invoking a domain-independent pre-rule that replaces an expression of the form  $P.geometry \text{ intersect } Q.geometry$ , where  $Q$  is a point, by an expression of the form  $Q.geometry \text{ in } P.geometry$ .

Note that the net effect of the pre-rule is to extend the semantics of **intersect** to cover other types of geometries. But, more importantly, this approach permits defining elaborated extensions that can be made context dependent, or even user dependent, simply by creating more sophisticated rules.

Other role we illustrate here is how a cooperative approach may help in the treatment of queries with ambiguous semantics, due to the presence of linguistic operators whose interpretation depends on context information. The example in this section follows a simple pattern: the semantics of an operator **op** is defined with the help of parameters whose values are fixed either in the application domain model, or in the user domain model. Moreover, the definition of **op** is expressed as a pre-rule that reduces **op** to other known operators. This approach is naturally adequate only when the semantics of **op** is ambiguous in so far as two users may have different parameter sets and, hence, may get different answers for the same query.

The following example deals with an imprecise metric operator, **near**, considered fundamental for the definition of spatial relationships. When a user formulates a spatial query, the notion of ‘near’ is naturally influenced by the context of the query and the user. Thus, for the purposes of our discussion, we consider the following definition for **near**:

$$(P \text{ near } Q) \equiv (\text{dist}(P,Q) < n * \text{length\_unit})$$

where **length\_unit** is a context dependent scalar parameter, and **n** is a scalar parameter. This definition of **near** is easily captured by the following example:

Consider the query ‘Retrieve all hospitals near school S1’:

```
select hospital.name
from hospital, school
where school.name = 'S1' and
```

To process this query, the system will first apply the pre-rule that defines *near*, which replaces an expression *P.geometry near Q.geometry* by an expression  $\text{dist}(\text{P.geometry}, \text{Q.geometry}) < n * \text{length\_unit}$ .

We can also define other imprecise operators (predicates), for example, the topological spatial operators *between*, *before*, *after*, etc.; and the temporal operators *soon*, *recent*, etc. The definition of these operators can be found in (Hemerly et al, 1993) and (Perez and Furtado, 1994).

Besides the roles above, the cooperative interface can cover other roles such as: (1) **Providing alternatives to a query:** For spatio-temporal databases, one way to generate alternatives to failed queries is by defining rules that relax search conditions based on spatio-temporal operators. In this case, if the alternative query succeeds, the system must of course let the user know that the original query failed, and that the answer corresponds to the alternative query; (2) **Explaining a failure:** The integrity constraints of a conceptual schema, as well as the semantic constructs of the data model adopted, are sources of rules that provide useful failure explanations to the user; (3) **Complementing queries:** this role means retrieving more information than the user explicitly asked for. The basic strategy is again to provide rules that modify queries, frequently by the inclusion of additional attributes. However, care must be taken to avoid becoming over-cooperative, passing to the user information that is actually irrelevant for his purposes. In the spatio-temporal case, the interface should, in general, distinguish any geometry the user did not explicitly request from those that he did. æ

#### 4. Concluding Remarks

One aspect of cooperativeness does not analyzed in this article, it is the problem involving how to present the results of the queries. In (Hemerly et al, 1993) and (Perez and Furtado, 1994) this problem is dicussed.

Our experience suggests that cooperativeness can help ensuring the successful use of temporal databases, by easing the task of users who could otherwise be overwhelmed by the complexity that tends to burden the interfaces to more powerful systems. It is important to note that one can begin with a minimal cooperative module enabling the rule-driven algorithm to intercept query requests, provided that the TDMS allows to represent some form of rules. The rules themselves can be limited in number and in their effect in terms of database searches, as required by cost versus utility considerations. Finally, as the database / knowledge base environment becomes richer, more rules can be gradually added to take advantage of the new possibilities. æ

#### References

- Casanova, M.A. and A.L. Furtado (1990); An information system environment based on plan generation; Proc. International Working conference on Cooperative Knowledge Based Systems, Univ. Keele, England (pp. 64-66)
- Chu, W., Chen, Q. and R.C. Lee (1990); Cooperative query answering via type abstraction hierarchy; Proc. International Working conference on Cooperative Knowledge Based Systems, Univ. Keele, England (pp. 66-69)
- Cuppens, F. and R. Demolombe (1989); Cooperative answering: a methodology to provide intelligent access to databases; Proc. 2nd International Conference on Expert Database Systems (ed. L. Kerschberg); Benjamin/Cummings (pp. 621-643)
- Egenhofer, M.J. (1989); Spatial Query Languages; PhD thesis, University of Maine, Orono, ME
- Frank, A.U., Kuhn, W. and Egenhofer, M.J. (1992); Geographic databases: The issues and some of the solutions; 5th International Symposium on Spatial Data Handling. Tutorial.
- Hemerly, A.S., Casanova, M.A. and A.L. Furtado (1991a); Cooperative behaviour through request modification; Proc. 10th International conference on Entity-Relationship approach, San Mateo, CA, USA (pp. 607-621)
- Hemerly, A.S., Casanova, M.A. and A.L. Furtado (1991b); Exploiting user models to avoid misconstruals; Proc. 1st International Workshop on Nonstandard Queries and Answers approach, Toulouse, France
- Hemerly, A.S., Furtado, A.L. and M.A., Casanova (1993); Towards Cooperativeness in Geographic Databases; Proc. DEXA.
- Navathe, S.B. and R. Ahmed (1989); A Temporal Model and a Query Language; Information Sciences, VOL. 49 (pp. 147-175)
- Pérez-Alcázar, J.J. and A.L. Furtado (1994); Towards Cooperativeness in Temporal Databases; Proc. Brazilian Computer Congress. (submitted)
- Webber, B.L. (1986); Questions, answers and responses: interacting with knowledge base systems; On knowledge base management system (ed. Brodie, M.L. and J. Mylopoulos), Springer-Verlag, Berlin