Querying TSQL2 Databases with Temporal Logic

Michael H. Böhlen¹, Jan Chomicki², Richard T. Snodgrass³, and David Toman⁴

¹Dept. of Mathematics and Computer Science, Aalborg University
Fredrik Bajers Vej 7E, DK-9220 Aalborg Ost, Denmark, boehlen@iesd.auc.dk

²Department of Computer Science, Monmouth University
West Long Branch, NJ 07764, chomicki@moncol.monmouth.edu

³Department of Computer Science, University of Arizona
Tucson, AZ 85721, rts@cs.arizona.edu

⁴Department of Computer Science, University of Toronto
Toronto, Ontario M5S 1A4, Canada

Abstract. We establish an exact correspondence between temporal logic and a subset of TSQL2, a consensus temporal extension of SQL–92. The translation from temporal logic to TSQL2 developed here enables a user to write high-level queries which can be evaluated against a space-efficient representation of the database. The reverse translation, also provided, makes it possible to characterize the expressive power of TSQL2. We demonstrate that temporal logic is equal in expressive power to a syntactically defined subset of TSQL2.

1 Introduction

In this paper, we bring together two research directions in temporal databases. The first direction is concerned with temporal extensions to calculus-based query languages such as SQL (e.g., [GN93, NA93, Sar93]). The issues addressed include space-efficient storage, effective implementation techniques, and handling large amounts of data. This approach includes the consensus temporal query language TSQL2 [Sno95], whose practical implementations should be forthcoming.

The second direction is concerned with defining high-level query languages with logical semantics, e.g., temporal logic [TC90, GM91, CCT94]. The advantages of using logic languages come from their well-understood mathematical properties [GHR94]. Logic languages are easy to use and make algebraic query transformation possible [CT95]. For instance, temporal logic has been proposed as the language of choice for formulating temporal integrity constraints and triggers [Cho95, CT95, GL93, LS87, SW95].

The semantics of temporal logic queries is defined with respect to sequences of database states [GHR94]. In temporal databases we do not want to construct and store all the states explicitly. Instead, various proposals have associated with each fact a concise description of the set of points over which the fact holds, such as a period¹ [NA93, Sar90, Sno87, Tan86] or a finite union of periods.

¹ We use the term 'period' in this paper rather than the term 'interval' commonly used in temporal logic because the latter term conflicts with SQL INTERVALs, which are unanchored durations, such as 3 months.
We show here how to translate temporal logic queries into TSQL2, enabling the user to write high-level queries which will be evaluated against a space-efficient representation of the database. While translations of first order logic to SQL have been previously discussed [AHV95, VGT91], we know of no translations from temporal logic to a temporal query language.

We start with a discussion of the basic framework in Section 2. We define the syntax and semantics of the two languages in question, temporal logic and TSQL2. In Section 3 we give the mapping from temporal logic to TSQL2. We conclude the section with an example and the discussion of some implementation issues. Section 4 discusses the reverse mapping, thereby relating the expressive power of (a subset of) TSQL2 and temporal logic.

2 Basic framework

Before comparing temporal logic and TSQL2 we have to set up a formal framework suitable to both languages. Time is considered to be integer-like: linear (totally ordered), discrete, bounded in the past, and infinite in the future. Our approach can be adopted to other kinds of time, e.g., dense, rational-like time, although some details of the mapping may in that case be different. We also take the point-based view which is predominant in the context of temporal logic. This view means that the truth-values of facts are associated with individual time points (also called instants). We assume a fixed time granularity.

We will consider only valid-time, which relates when facts are valid in reality [JCE+94]. In particular, transaction time, which relates when facts are stored in the database, is not considered.

2.1 Temporal logic

Temporal logic is an abstract language, i.e., a language which is defined with respect to abstract temporal databases [Cho94]. An abstract temporal database, in turn, is a database which captures the formal semantics of a temporal database without considering representation issues.

It is possible to view an abstract temporal database in several different but equivalent ways. We choose here the timeslice view (called snapshot in [Cho94]) in which every time instant is associated with a (finite) set of facts that hold at it. For integer-like time, this view leads to an infinite sequence of finite database states \((D_0, D_1, D_2, \ldots)\).

Example 1. Table 1 presents an example of an abstract temporal database, viewed as a sequence of states. The database represents information about Eastern European history, modeling the independence of various countries [Cho94]. Each fact indicates an independent nation and its capital. This relation will be used as a running example throughout the paper.

---

2 An implementation of the translation from temporal logic to TSQL2 is publicly available at http://www.iesd.auc.dk/~boehlen/.