Abstract: Temporal logic is obtained by adding temporal connectives to a logic language. Explicit references to time are hidden inside the temporal connectives. Different variants of temporal logic use different sets of such connectives. In this chapter, we survey the fundamental varieties of temporal logic and describe their applications in information systems.

Several features of temporal logic make it especially attractive as a query and integrity constraint language for temporal databases. First, because the references to time are hidden, queries and integrity constraints are formulated in an abstract, representation-independent way. Second, temporal logic is amenable to efficient implementation. Temporal logic queries can be translated to an algebraic language. Temporal logic constraints can be efficiently enforced using auxiliary stored information. More general languages, with explicit references to time, do not share these properties.

Recent research has proposed various implementation techniques to make temporal logic practically useful in database applications. Also, the relationships between different varieties of temporal logic and between temporal logic and other temporal languages have been clarified. We report on these developments and outline some of the remaining open research problems.
3.1 INTRODUCTION

Time is ubiquitous in information systems. Almost every enterprise faces the problem of its data becoming out of date. However, such data is often valuable, so it should be archived and some means to access it should be provided. Also, some data may be inherently historical, e.g., medical, cadastral, or judicial records. Temporal databases provide a uniform and systematic way of dealing with historical data. Many languages have been proposed for temporal databases, among others temporal logic. Temporal logic combines abstract, formal semantics with the amenability to efficient implementation. This chapter shows how temporal logic can be used in temporal database applications. Rather than presenting new results, we report on recent developments and survey the field in a systematic way using a unified formal framework [GHR94; Cho94]. The handbook [GHR94] is a comprehensive reference on mathematical foundations of temporal logic.

In this chapter we study how temporal logic is used as a query and integrity constraint language. Consequently, model-theoretic notions, particularly formula satisfaction, are of primary interest. Axiomatic systems and proof methods for temporal logic [GHR94] have found so far relatively few applications in the context of information systems. Moreover, one needs to bear in mind that for the standard linearly-ordered time domains temporal logic is not recursively axiomatizable [GHR94], so recursive axiomatizations are by necessity incomplete.

Databases are inherently first-order structures and thus in this chapter we are primarily interested in first-order temporal logic. This is in sharp contrast with another major application area of temporal logic, program verification, where the formalisms studied are usually propositional [MP92; Pnu86].

We introduce here a number of fundamental concepts and distinctions that are used throughout the chapter. First, there is a choice of temporal ontology, which can be based either on time points (instants) or intervals (periods). In most database applications the point-based view is more natural and thus we concentrate on it in this chapter. However, in section 3.5 we briefly discuss interval-based temporal logic. (Intervals are predominant in AI applications.) Second, time can be single-dimensional or multi-dimensional. Multiple time dimensions can occur if, for example, multiple kinds of time (e.g., transaction time and valid time [SA86]) are required in an application. In addition we show that multiple temporal dimensions are necessary to evaluate general first-order temporal queries. Except for section 3.5, we adopt the single-dimensional view. Finally, there is a choice of linear vs. nonlinear time, i.e., whether time should be viewed as a single line or rather as a tree [Eme90], or even an acyclic graph [Wol89]. Although nonlinear time is potentially applicable to some database