Overview and Directions to the Reader

The contributions in this publication are divided into a brief introduction to temporal database research, followed by the five parts already mentioned in the preface. In addition to the general overview provided here, each of these parts begins with an overview that in more detail motivates the part and outlines the contents of its chapters. Each part contains chapters that offer quite extensive surveys of related research, and the individual chapters are generally self-contained.

The very first chapter offers a brief introduction to temporal database research and has been written to provide a context for the remaining chapters. It concisely defines fundamental concepts, illustrates the diverse challenges faced in temporal database research, samples some of the contributions, and offers a look into the future of temporal database research. Although most chapters are self-contained, it is recommended that the reader unfamiliar with temporal databases studies this chapter first.

Part I is concerned with the *semantics of temporal data* and offers a conceptual foundation for the remaining parts. It contains seven chapters, each of which is self-contained. The opening chapter offers a survey of a good deal of the research presented in the remaining chapters of this part.

Most temporal database research focuses on two fundamental and orthogonal temporal aspects of data, namely valid time and transaction time. The valid time of a fact records when the fact is true in the modeled reality, and the transaction time of a fact records when the fact is stored as current in the database. These aspects, also termed dimensions, of data are of interest in a wide range of applications. The first chapter delves into the meaning of valid time and transaction time, thus mapping out the specialized interrelations that may exist between these aspects of data in concrete applications and exploring the implications of the interrelations; it also studies how multiple valid times and transaction times may be associated with the same data.

The next two chapters offer a comprehensive study of the semantic implications of storing the current time, captured by the variable *now*, in databases. The first chapter presents a framework for defining the meanings of temporal databases containing the variable *now* and for querying such databases, and the second chapter is devoted to the issues that relate to the modification of *now*-extended databases.

The chapter, “Unifying Temporal Data Models,” introduces the Bitemporal Conceptual Data Model and positions this model at the center of a (point-based, in the terminology of the next chapter) framework of well-behaved transformations that bridges the gaps between existing temporal data models and their algebraic query languages. The next chapter most prominently formalizes what it means for a query language to “respect” the time intervals that are often associated with data, thus offering an intermediate between the interpretation of time intervals as abbreviations of sets of points and as non-decomposable values. When capturing the transaction time of data, deletion statements only have a logical effect. The last chapter introduces and studies physical deletion, termed vacuuming.

Part II presents the core data model and query language of the temporal TSQL2 query language, which extends the formally standardized SQL-92 query language. This part represents a single, integrated *case study in temporal query language design*. Its fourteen chapters stem from a unique and intense one-year initiative aimed at defining a consensus temporal query language—based maximally on past contributions by the community—which could serve as a foundation for future research and as a recommendation to vendors interested in offering database products with built-in temporal support.

The reader will notice that this part is based quite solidly on the conceptual foundation of the previous part and that TSQL2’s data model is based on the Bitemporal Conceptual Data Model. It is recommended that the reader initially studies the short tutorial in the part’s first chapter. In addition, it is beneficial to be familiar with the material offered in the chapters, “The TSQL2 Data Model” and “Valid-Time Selection and Projection,” before proceeding to subsequent chapters. The penultimate two chapters are not part of the language design per se, but argue for the feasibility of the language design by offering a foundation for its implementation. The last chapter supplies the language specification in the format of the SQL-92 standard.

Part III, which also concerns *temporal data models and query languages*, describes results of research that continued where TSQL2 ended. With only one year available and with eighteen researchers involved, the initiative that brought about TSQL2 was hectic, and not all aspects of the language design received adequate attention. The first chapter in this part identifies some of the trouble spots and establishes a foundation for new requirements, which point to an alternative design. The next chapter introduces a temporal notion of upward compatibility that a temporal extension of a query language and data model should satisfy, it proceeds to study the implications of this requirement, and it evaluates all existing temporal SQL extensions with respect to upward compatibility and temporal upward compatibility.

The first two chapters set the stage for a better successor to TSQL2. The next trio of chapters propose one such successor—they configure the part, SQL/Temporal, of the now almost completed successor to the SQL-92 standard. The reader interested in the background of the design requirements to the proposal should study the first two chapters of Part III before proceeding. The trinity's first chapter provides an overview of the proposal and is a natural read for those interested in understanding this successor to TSQL2. The next two chapters comprise the actual expert contributions submitted to the standards bodies. The first configures SQL/Temporal with support for valid time, and the next adds support for transaction time. Together, they flesh out the picture initially drawn by the first chapter, and they also encompass specifications of the temporal support in the formalized language chosen by the standards bodies.

The final chapter puts forth another successor to TSQL2, termed ATSQL. No standardization process has affected this language, making it perhaps cleaner and more "theoretical." In comparison to the SQL/Temporal proposal, ATSQL offers better built-in support for formulating temporal query language statements, as well as support for queries that are simultaneously interval and point-based, and it has a formal, denotational-semantics style definition. On the other hand, it assumes a set-based framework, thus avoiding the issues associated with duplicates, and it is less rich in features.

The source code of an implementation of the core of the SQL/Temporal proposal as well as an implementation of a core subset of ATSQL are publicly accessible on the Web. Barring the provisos stated above, all chapters of this part are self-contained and may be read in isolation.

Part IV follows up on the coverage of data semantics and data models and query languages by considering the *design of temporal databases*. The first two of its five chapters concern logical design in a relational database context and are independent from the last three chapters, which concern conceptual design in an Entity-Relationship context.

The first chapter observes that a rich theory exists for database design in the conventional relational model, which is, however, not readily applicable to temporal relational data models because these latter models entertain new kinds of relation structures. Following an investigation of all existing temporal normalization concepts, the chapter adopts a point-based approach and generalizes existing normalization theory—dependencies, keys and normal forms—to apply to all temporal relational data models that possess time-slice operations, which almost all temporal models do. The second chapter takes a next logical step: it provides concepts for capturing additional, temporal semantics of data and then proposes additional guidelines for decomposing temporal relations, and also standard relations, based on their temporal semantics, and in this way, completes the picture.

The remaining three chapters of the part aim to temporally enhance the design of databases at the conceptual level and are based on the ER model. It is common practice to design a conceptual database schema using a graphical tool that supports some variation of the ER model and then to generate a relational database schema from the ER diagram. The first chapter makes available a comprehensive survey and evaluation of the existing temporal ER models. Observing that no single model is entirely satisfactory, the second chapter presents the design and semantics of a new temporal ER model, TimeER, based on an ontological foundation and explicit requirements. This second chapter should be read before proceeding to the last chapter, which gives detailed mappings from TimeER to a surrogate-based relational model and the regular relational model.

Part V investigates the *implementation of temporal data models and query languages*. The first chapter proposes a new technique for the efficient computation of the operation that correlates related data stored in different valid-time relations. This operation is fundamental when querying temporal data, and existing systems apply a brute-force technique to computing this operation, which is often inefficient. The next chapter offers techniques for the efficient computation of another fundamental operation, namely the operation that retrieves the state, current at a particular time, from a relation that supports transaction time.

The two chapters that follow give complementing techniques for indexing bitemporal data, which has both a valid-time and a transaction-time dimension. Because of the special variable *now*, bitemporal data is unlike any other data, and all existing indexing techniques fall short in supporting this data. The first technique generalizes the R-tree, an index proposed for spatial data, to contend with regions that grow continuously as time advances. The second technique elects to subject the bitemporal data to a number of transformations that render all regions static. This in turn makes it possible to index the transformed data using a total of four conventional R-trees. Queries that are transformed to counter the data transformations achieve perfect precision and recall.

The last three chapters all follow from the same motivation and consider different aspects of the same general challenge. Using a conventional database management system (DBMS) architecture for developing a temporal DBMS is a daunting task that may only be realistic for major database vendors with access to an existing commercial code basis. A layered architecture may be an effective alternative: temporal support is achieved by a layer that is effectively an advanced application built on top of an existing DBMS. In this architecture, temporal SQL statements are translated into SQL statements understood by the underlying DBMS, the services of which are reused. The first chapter introduces concepts and techniques central to a layered implementation approach. The next chapter offers a taxonomy of different layered architectures, evaluates their strengths and weaknesses, and also gives examples of their use. The last chapter is devoted specifically to the interaction between the timestamp values placed on data and the support for well-behaved user transactions. Without a careful choice of timestamp values, transactions result in anomalous database states. The chapter delivers techniques that yield simple, consistent, and efficient support for modifying bitemporal databases in the context of user transactions.

In summary, the five parts of the publication cover four topics: the semantics of temporal data, the design of data models and languages for temporal data, the design of databases expressed in terms of temporal data models as well as temporally enhanced design of conventional databases, and the effective and efficient implementation of temporal data models and languages.