1. OVERVIEW

Real-time concurrency control has been an active research topic in the past decades. It has been recognized by a number of researchers that the notion of serializability is too strict a correctness criterion for concurrency control in accessing real-time data. In avionics software, for example, the precision of an answer to a query involving sensor data is often acceptable as long as the data is sufficiently timely, even though updates are sometimes performed in violation of the usual serializability criterion. Instead of read/write locks, a “cyclic executive” is also routinely used in these applications to enforce a set of timing constraints on data access which in turn guarantees data consistency, the usual database locking protocols being too inefficient for the purpose. Obviously, violation of serializability must be justified in the context of the semantics of the application domain.

Three increasingly restrictive criteria for correctness are commonly accepted and have been studied in depth. They are: final-state serializability, view serializability, and conflict serializability [17]. In recent years, other different correctness criteria and related concurrency control algorithms have been proposed for different purposes and application. The relaxation of correctness criteria not only helps the deployment of (real-time) database applications in various environments and application domains, such as distributed environments, but also provides the performance improvement which is otherwise impossible for many real-time applications. In other words, relaxed correctness criteria not only provide more functionality, e.g., queries of different precision levels under Epsilon-serializability [6, 19, 20], but also relax the constraints in transaction processing, such as similarity-based concurrency control [7, 8].
Kuo and Mok [7] explored the similarity of data to provide a semantic foundation for accessing data in a real-time fashion. Similarity is a binary relation on the domain of a data object. Every similarity relation is reflexive and symmetric, but not necessarily transitive. In a schedule, we say that two event instances are similar if they are of the same type (read/write) and access similar values of the same data object. The concept of similarity can be used to extend the usual correctness criteria for transaction scheduling [7] and to derive flexible and efficient concurrency control algorithms in uniprocessor, multiprocessors, and even mobile environments [8, 9, 12, 13]. The results justify the weaker notion of correctness that has been employed on an ad hoc basis in many real-time applications where the state information is "volatile" and the value of data depends on its timeliness. A distributed real-time data management interface was also built on an Intel multiprocessor machine [10]. Xiong et al. [22] further exploited temporal data similarity and evaluated the performance improvement of transactions when combinations of similarity and forced wait policies were considered. A force wait policy may force a transaction to delay further execution until a new version of sensor data becomes available.

Peng and Lin proposed the idea of compatibility matrix to allow transactions to acquire different degrees of consistency requirements [18]. Their work was motivated by avionic systems and automated factories that have a limited number of high-speed sensors with frequent user-initiated command processing. The rationale behind their work was that the consistency between the device readings and the current values used by transactions could be more important than the serializability of transactions. DiPippo and Wolfe [2] also proposed object-oriented techniques to support both logical and temporal consistency requirements. Compatibility functions of objects were proposed to allow the tradeoffs among these requirements.

Epsilon-serializability (ESR) [19, 20] formalizes the query behavior by deriving the formulae that express the inconsistency in the data values read by a query. Transactions are associated with limits of importing inconsistency and exporting inconsistency. Query transactions are allowed to view inconsistent data in a controlled fashion. Kamath and Ramamritham [6] then introduced the idea of hierarchical inconsistency bounds that allows inconsistency to be specified at different granularities such as transactions and objects. They provided mechanisms to control the inconsistency and reported the evaluation of the performance improvement due to ESR.

There are also many weak correctness criteria being proposed for non-real-time transaction systems [14, 15]: In particular, Garcia-Molina and Wiederhold in [5] discarded consistency considerations for read-only transactions, with the stipulation that, after read-only transactions have been removed, the resulting schedule should be serializable. Garcia-Molina and Salem [4] proposed the