The API Specification for Real-Time Database Systems

NING Wei¹, WANG DAO-lin²

¹ School of Geodesy and Geomatics, Huaihai Institute of Technology; Lianyungang 222001, China
Email: ningwei2380@163.com
² Department of Computer Science and Technology, Taishan College; Taian271000, China

Abstract — The commercialization is a critical step for further research and development of RTDBS (Real-Time Database Systems). Thereinto, Application Programming Interface (API) is an important factor concerned by developers and end users in commercial RTDBS products. However, few or none has paid attention to the API of RTDBS. In this paper, conformance classes and related API of RTDBS are discussed, based on a real-time object-oriented database model. Especially, event mechanism is proposed to meet requirements on active pursuit of timely transaction processing.

Index Terms — real-time database system, event mechanism, application programming interface, conformance classes

I. INTRODUCTION

A Real-Time Database System (RTDBS) can be viewed as an amalgamation of a conventional Database Management System (DBMS) and a real-time system. Like a DBMS, it has to process transactions and guarantee ACID properties. Furthermore, it has to operate in real-time, satisfying time constraints imposed on transaction commitments. A RTDBS may exist as a stand-alone system or as an embedded component in some time-critical systems [1]. Example applications include telephone switching, radar tracking, automated manufacturing, patient monitoring, and so on.

Generally, there are three distinguishing features in RTDBS from traditional DBMS: requirements on temporal consistency of some data, requirements on timing constraints on transaction executions, and requirements on active pursuit of timely transaction processing. Large amount of research has been made for satisfying requirements mentioned above, including real-time transaction scheduling, concurrency control, indexing, I/O scheduling, and so on. But few have paid attention to the Application Programming Interface (API) of RTDBS. However, API of a RTDBS is an exhibition of the functionalities and a key component used directly by developers. With the appearance of commercial RTDBS products such as EagleSpeed [2], Clustra [3] and Agilor [4], developers of time-critical systems attempt to solve some problems in handling large amounts of data and stringent timing constraints via RTDBS. But developers are troubled with proprietary API of different RTDBS when they try to transplant their applications from a RTDBS to another.

In fact, the initial effort on the standardization of RTDBS interface was made by the U.S. Navy's Next Generation Computer Resources (NGCR) Database Interface Standards Working Group (DISWG) [5]. The work of the DISWG was to generate a list of requirements so as to establish interface standards for use in mission-critical systems. Next, J. PRICHARD extended SQL to define RTSQL for supporting RTDBS based on the DISWG requirements [6]. Although RTSQL includes necessary extensions such as temporal consistency constraints on data, timing constraints on transactions, it is rarely adopted in RTDBS prototypes or commercial RTDBS products. There are several limitations to facilitate the use of GRDBS via RTSQL. One is that RTSQL lack in conformance classes so that RTSQL cannot adapt to different environments through reducing it. Sometimes the preprocessor of RTSQL is cumbersome, especially for embedded applications.

This paper aims is to make an effort to establish a standard API specification for RTDBS, as an imperative means of manipulating the database more efficiently than RTSQL, besides ensuring real-time performance. The rest of this paper is organized as follows. Section 1 discusses system philosophy of RTDBS, including data model, transaction model and conformance classes. Section 2 illustrates necessary API provision by functionality categories. Section 3 summaries our work and put forward the future work.

II. SYSTEM PHILOSOPHY

Generally, RTDBS acts either as a stand-alone system or as an embedded component in applications. The system model of RTDBS and application can be showed as follows:
The different system models and application requirements may impose different functionalities and interfaces on the RTDBS. As is well known, functionalities and performance contains contradictory elements. Especially for mission-critical applications, the best choice is to exchange extra functionalities for better performance. However, developers expect a consistent interface to different RTDBS or their different versions. This section will introduce a model of RTDBS as a conceptual basis for constructing the API specification of RTDBS, and then establish conformance classes of API. The model of RTDBS is composed of data model and transaction model referred on the framework for real-time object-oriented database models proposed by Juha Taina and Sang H. Son [7].

A. Data Model

The data model concerns how to structure data logically. The object-oriented data model is highly recommended for RTDBS for the fact that a lot of real-time applications need data models that are richer than the traditional used relational data models, from the features including extra modeling power, method-based access to objects, relationships between objects, embedded integrity constraints, and tightly integrated with the host language such as C++, C, Smalltalk, or Java. Although a real-time object-oriented model is used, it won’t affect the universality of the proposed API specification. The methods of objects can be implemented as basic operations of API.

B. Transaction Model

The transaction can be defined as a eight-tuple <N, A, O, C, PR, PO, R, E>, where N is a unique name or identifier, A is a set of transaction attributes including type, deadline, period, criticality and read/write mode, O is a set of operations, C is a set of operation constraints, PR is a precondition, PO is a postcondition, R is the result of the transaction, and E is a set of events which will be triggered when the violation on a constraint of C occurs. The constraint C is used to define the properties such as precedence constraints, execution constraints, resource limitations, gauge and so on. The precondition consists of requirements that define the conditions that the transaction must satisfy to begin execution. The postcondition contains predicates that define what constitutes a correct execution of the transaction.

The inflexibility of the conventional ACID properties often causes conservative execution of transactions that reduces real-time performance. To better support real-time applications, RTDBS will deploy flexible transaction structure with relaxed ACID properties. The flexible transaction structure allows developers to select the degree of consistency and correctness they need from the database.

C. Active Property and Events

For time-constrained applications, it is important to monitor the constraints defined on data, transactions and database. Once the specified constraints are violated, the system will inform applications by events as defined in data model and transaction model.

Event, as an effective way for RTDBS to provide active feature, can be easily supported by RTDBS API.

On the other side, event mechanism is imperative for asynchronous execution of transactions, especially for long-term transactions. It can avoid applications idly waiting for the finish of a transaction. Furthermore, periodic or event-triggered transactions must to be event-driven.

D. Conformance Classes

In many cases, RTDBS is used in embedded environments. One of the most important features in embedded environments is diversity, including the diversity of hardware and software platform. Various requirements from the applications and various capabilities of specific platform (e.g. processor, memory) demand different features and capacities of RTDBS.

The conformance classes and their relationship are showed in Figure 2.

- Single user DB. This version only allows one user thread to access the database at a time, thus removing the need of supporting concurrency control, transaction and event mechanism.
- Multi-user DB. This version allows multiple user threads to access the database simultaneously, but does not support concurrency control and transactions. The multi-user database is suitable for applications that have only one update but multiple read-only operations on a data object simultaneously. The difference between the multi-user DB and the single user DB is that multi-user DB can deploy event mechanism, in which one thread execute operations and another thread wait for handling related events from the database. Although there are no transactions-related events in the multi-user DB, the object-related events can be provided by the database and be used by the applications.
- Transactional DB. The transactional DB supports transaction and concurrency control. This version of RTDBS can be used in applications where there are multiple read and write operations on a data object at a time. Such applications need RTDBS providing transactions and concurrency control mechanism to
guarantee the ACID properties and real-time requirements. So that the transaction related API is necessary.

The relationships of the API provision and conformance classes are listed in the table below. The API of data manipulation is necessary for all of the three versions of database. The API of event notification API is provided by the multi-user DB and transactional DB, while the API of transaction management only appears in the transactional DB.

<table>
<thead>
<tr>
<th>API</th>
<th>Data Manipulation</th>
<th>Event Notification</th>
<th>Transaction Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single User DB</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Multi-user DB</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Transactional DB</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

III. API PROVISION

In general, RTDBS will pre-allocate related resources at startup so as to achieve better predictability. The operations of database management will not be discussed since we focus on active pursuit of timely transaction processing. The API of data manipulation, event notification and transaction management and their optional features will be explained according to the conformance classes.

A. Data Manipulation

The group of API is used to manipulate the data objects in database. The data manipulation operations can be done within transactional DB or without a transaction in single user DB or multi-user DB, which consists of data definition operations and data retrieving operations.

The data manipulation method can be defined as a five-tuple $<N, O, C, T, A>$, where:

- $N$ is a name of the method, which denotes the functionality of the method such as create, add and set operation on an object;
- $O$ is an object name, on which method $N$ will execute the specific operations;
- $C$ is an optional parameter, which indicates how method $N$ supports temporal constraints. There are three options explained as follows:
  - Leave out of consideration on absolute and relative temporal consistency constraints on objects.
  - Only support constraints of absolute temporal consistency on an object. Once the object is outdated, the related method of the object in TM will be invoked to update data forcefully.
  - Support not only absolute temporal constraint on an object, but also relative constraint between a set of data objects.
- $T$ is an identifier of the transaction in which the method is executed.
- $A$ indicates execution model of the method, i.e. synchronous or asynchronous execution.

B. Event Notification

Event mechanism is imperative to support asynchronous execution and control execution process of transactions. Once an exceptions occurs, application system can response timely via event notification. At the same time, RTDBS may provide an appropriate manner of supporting application systems to subscribe their interested events. The identifiers of objects or transactions and event types need to be included in event structure, so that applications can know what had happened exactly. The API of event notification also provides method of defining event handler for an event of a set of events. The event handler is a callback function, which will be invoked once a specific event occurred.

C. Transaction Management

In transactional DB, a transaction can be considered as a series of methods of data objects in the context of RTDBS. The methods of data objects may be read-only, write-only, or read-write, thus a transaction can be defined as read-only, write-only, or read-write. Furthermore, transactions have different criticalities and deadlines, thus be endowed with different priorities.

Some transactions can be pre-defined in RTDBS, which may be periodic or event-triggered. When implementing the API of transaction management, there are several options about timing constraints on transactions to be considered as follows:

- Deadline-only, that only absolute deadline on transactions can be supported.
- Period-allowed, that periodic transaction can be scheduled automatically according to their periods.
- Full-support, that all types of transactions can be supported, including event-triggered transactions with preconditions, relative deadlines, postcondition, and so on.

The API of transaction management facilitates developers to manage a specific transaction directly, which includes the following operations:

- Create/delete a transaction. When a transaction is
created, developers can define its related operations and properties. Subsequently, transactions can be scheduled according to their priorities.

- Start/Commit/Abort a transaction. These operations are used to identify a transaction completely.

IV. SUMMARY AND ISSUES

The conformance classes and API provision of RTDBS will be beneficial to RTDBS vendors and developers. In fact, this paper summarizes our work on an embedded real-time database for smart phones, which is a research and development project funded by France Telecom. The conformance classes of RTDBS may be rough. However, it is in favor of adapting RTDBS to various application requirements and to be implemented or upgraded.

The proposed event mechanism and asynchronous execution of transaction allow user threads work on other tasks while RTDBS executes their requests. And developers can select the degree of consistency and correctness they need from the database, which can improve the data availability and real-time performance.

In the future, there should be more effort on the standardization of RTDBS and related interfaces. And how to retrieve historical data of temporal objects effectively is another issue need to be resolved.

REFERENCES