A Survey on Real Time Database

Krishna Shah[1], Devansh Jani[2]
Department of Computer Engineering,
Institute of Technology, Nirma University
17mcec15@nirmauni.ac.in[1],17mcec07@nirmauni.ac.in[2]

Abstract

Real time database supports variety of applications as per their performance and response time. Data management in Real time system has been maintained to produce fast turnaround query time. In this paper will explain what are real time database applications and its characteristics in respect with data and its consistency. It further explains transactions and scheduling. Policy to sustain against deadlock is explained and measures that results in to concurrency are explained and comparison is shown.

Keywords

Real−time,Database,Acidproprieties,Data,Consistency,absolute−validity,Relative−consistency, Deadlock,Transaction,Concurrency,Scheduling,Pessimistic,Optimistic,Speculative

I. Introduction

Many database needs large amount of storage which must provide features and process this data to get successful operation and for such large data we need database management system. Real time database is a database system that handles the work-load with constantly changing data. They are dynamic in nature and used for real time computation such as banking,stock market and many more.
Some Characteristics of real time system can be as Time Constraints,Deadlines, Correctness Criterion, Safety-Criticality, Concurrency, Task Criticality,Custom-Hardware, Stability, Exception-Handling.

A. Basic Model of Real time System

The Figure 1 shows the basic model of real time system in terms of blocks. It has sensors which converts physical characteristics of its environment into electrical signals.Actuator is a device which inputs from output interface of computer and converts them in to electrical signals. Interface unit is normally commands received from the CPU and are delivered to actuator The interface takes care of buffering and handshake control.

II. Real Time Database

Real-Time Database System are database systems that are made to satisfy certain real time system constraints such as time and priority.Factors to tune traditional DB:
Schedulability: The ability of tasks to meet all hard deadlines.

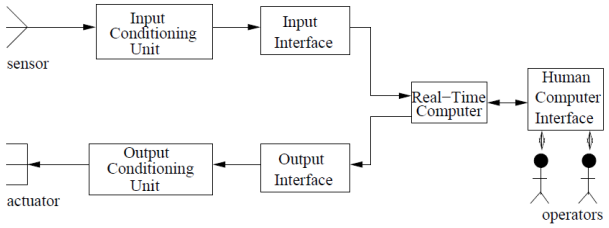

Fig. 1. Basic Model of Real time System

Response time: The worst-case system response time to events **Stability in overload:** The system meets critical deadlines even if all deadlines cannot be met **Timing constraints:** e.g. deadline is a timing constraint associated with the transaction. **Criticalness:** It measures how critical it is that a transaction meets its timing constraints. Criticalness is a different concept from deadline as a transaction may have a very tight deadline but missing it may not cause harm to the system. **Resource requirements:** Number of I/O operations to be executed. **Expected execution time:** Estimate or experimentally measured value of worst case execution time. **Data requirements:** Read sets and write sets of transactions. **Periodicity:** If a transaction is periodic, what its period is. **Time of occurrence of events:** In witch point in time a transaction issues a read or write request.Other semantics - Transaction type (read-only, write-only, etc.).

Some examples of real time database are Telecommunication systems they are firm real time databases, routers and network management system database are also included in the same category. Control systems in nuclear power plants are an example of Hard real time database system, as late retrieval of critical data can cause hazardous effects. Multimedia servers for real-time streaming are based on soft real time databases even if a frame is not retrieved or retrieved late it won't affect the performance much.[3] E-commerce and e-business, stock market and financial services use hard real time databases as there is a possibly of a huge monitory loss. Web-based data services use soft real time databases.

Response-Time Predictability has some Hurdles,such as Blocking and transaction, Requirement to meet the ACID properties, Unpredictability of disk access time and page faults, Data dependency of transaction executions, Dependence of the transaction's execution sequence on data values Data and resource conflicts, Dynamic paging and I/O, Transactions abort and the resulting rollbacks and restarts, Communication

delays and site failures on distributed databases [8]

Techniques to improve response time: Use main memory database, Need worst-case predictability, Use real memory addressing, Best effort in scheduling,

ACID Properties **Atomicity:** A Transaction is done either completely or not at all(Partial transactions are allowed) **Consistency:** Transactions are executed in a given sequence(Aperiodic, sporadic) **Isolation:** The actions of a transaction are not visible to any other transactions until it is committed(Not required!) **Durability:** The actions of a database are permanent(Not all real time data is durable) In real-time databases, relaxing ACID depends on application semantics.[2]

They differ in three ways majorly focus on : **1)Temporal data** Temporal data is type of data who's validity is lost after a predefined time interval. Either the data is updated with newer version of data or the data is archived or discarded. An example of temporal data is temperature sensing service with multiple sensor nodes which send temperature information at specified time intervals. As soon as new data is retrieved the old temperature data is discarded modifying the RTDB. Another example of Temporal Data is Bit-coin price database. As soon as new price quotations come in, data of the previous prices becomes obsolete[5]. **2) Timing Constraints on Database Operations:** Transactions and tasks are very similar. They both are scheduled similarly and are basically a unit of work. But a differentiating factor is that a transaction requires data to be in exclusive mode compared to a real time task, moreover a real time task could be predictable while a real time database operation is not a predictable operation as there are multiple dependencies on the executing transaction. **3) Performance metric:** In a traditional database system number of multiple concurrent transactions matter the most, while in real time database response time matter the most.

A. Real-Time Database Application Design Issues

Time taken to fetch data from secondary storage device is much more than time to fetch data from primary memory or cache. This makes it difficult to predict predict the response time. Another important factor is rollback and dependencies of on transaction on another, these two factors also affect the predictability of a transaction.[7]

B. Data and Consistency

Maintaining logical consistency of a database is not only an important task RDBMS has to maintain temporal as well as timing properties of a transaction. Temporal consistency has two requirements to be satisfied they are as follows:

1) Absolute Validity: It is valid only between absolute points in time. It's used to maintain consistency of environment with real time database. Denote a temporal data item in RTDB by d: (value, avi, timestamp) Here, dvalue denotes the current value of d, dtimestamp denotes the time when the d was updated, davi denotes d's absolute validity interval, i.e., length of time interval following dtimestamp during which d is considered to have absolute validity.[4] Condition for Absolute Validity: A data item d is absolutely valid, iff (Current time | dtimestamp) $<$ davi

2) Relative Consistency: Relative consistency is used to derive another dependent data from temporal data. Assume a data d in R (relative consistency set) d has a correct state if dvalue is logically consistent – satisfy all integrity constraints d is temporally consistent relative consistency: For arbitrary d' in R, $|\text{dtimestamp} - \text{dFLtimestamp}|$ Rrvi Each R is associated with a relative validity interval (rvi)

Example: Relative Consistency Let us assume that the current fuel level is 30 litres at time 100 msec and the distance travelled is 10 kilometers. Also assume that the avi is 20 msec. dfuel (30 litres, 20 msec, 100 msec) ddistance (10 kms, 20 msec, 110 msec) We read these two values from two different sensors. So, there will be a difference or delay in data arrival at the controlling system. If rvi is 15 msec for example, then the given data are relatively consistent. [4]

III. Transactions in Real-Time Database System

A. Scheduling in Transaction Processing

Hard deadline transactions: Cause catastrophic results if missed.
Soft deadline transactions: Does not cause havoc but system performance degrades.
Firm deadline transactions It causes undesired results, but they are not as catastrophic as Hard Deadline systems.

B. Types of transactions:

Write only: These are the transactions which write the state of environment into the real time database. Update: These transactions retrieve new data items from controllers or users and update the existing rel time database. Read only: These transactions fetch data items from the existing real time database and provide them to the users or controlling system. Problems encountered in real time transactions:
wasted restart: When a high priority transaction kills a low priority transaction and later at some time the high priority transaction fails missing its deadline, this is called as wasted restart. [1]
wasted wait: It occurs when a low priority transaction waits for a high priority transaction to commit and finally after some time the high prior transaction is killed as it misses it's deadline.
Wasted execution: It occurs in validation phase when a low priority transaction is restarted because of an unfinished higher priority transaction. Two phase is the method which suffers the most due to the problem of wasted restart and wait, while the Optimistic methods suffer due to unnecessary restart and wasted execution. [1]

C. Scheduling in Transaction Processing

Static table-driven approaches: Static table-driven approaches are scheduled at the very beginning and a explicit schedule is constructed, but are used at run time to start task execution.

Static priority-driven preemptive approaches: Tasks are ran on the basis of highest priority first. No schedule is made in Static priority-driven preemptive approach.

Dynamic planning-based approaches: Feasibility is the main criteria in this approach. At the run time feasibility of a transaction is checked and accordingly it is executed. If not found feasible the transaction is discarded.

D. Scheduling Deadlocks

Deadlocks occur even in real time databases. Deadlocks are a very hazardous condition in real time databases as they could delay critical transactions. Aborting a transaction is also a very costly operation considering real time databases. Five deadlock resolution policies for real time database: **Policy 1:** Abort all the transactions invoking deadlock detection. **Policy 2:** Abort the transaction causing the delay in execution(tardy transaction). If there does not exists any delay causing transaction, abort the furthest deadline transaction. These operations could be completed using tracing deadlock cycle **Policy 3:** Abort the transaction causing the delay in execution(tardy transaction). If there does not exists any delay causing transaction, abort the earliest deadline transaction. These operations could be completed using tracing deadlock cycle **Policy 4:** Abort the transaction causing the delay in execution(tardy transaction). If there does not exists any delay causing transaction, abort the least critical transaction. These operations could be completed using tracing deadlock cycle **Policy 5:** Abort the transaction which is not feasible and is also non critical. If transactions are feasible then abort the non critical transaction. Remaining execution time of the transaction must be known. And total execution time of each transaction should be known at the beginning.

IV. Concurrency Control

Some of the known concurrency Control Protocols are as mentioned as follows. Validation, Locking, Multiversion, Time-stamping. They all are created to enforce serializability. Although these protocols are good working for traditional systems, they need to be modified a little, to prevent, detect and respond to collisions. [4] Concurrency Control in Real-Time Databases can be obtained from the following methods: Pessimistic Methods, Timestamp Ordering Methods, Serialization Graph Testing, Locking Methods, Optimistic Methods, Backward Validation Methods, Forward Validation Methods and Serialization Graph Testing.

A. Pessimistic Concurrency Control

When concurrent operations are performed on real time database care should be taken that locks are used properly. Some of the locking mechanism used in RTDB used are: **Two-Phase Locking (2PL)** Priority is given to the lock operation and locking operation are performed before unlock operation in shrinking or expanding phase. 2PL still suffers from deadlocks. Priority inversion could be use. Two-phase locking + A Priority Assignment Scheme, such as RM or EDF is used.

Locking Methods are based on 2PL :
2PL Wait Promote
2PL High Priority
2PL Conditional Priority Inheritance
Priority Ceiling Protocol

2PL Wait Promote: The shortcomings of the pure 2PL protocol.TR is the requesting transaction and transaction TH is holding the data item.
/* pri(T) denotes priority of transaction T*/

if pri(TR)> Pri(TH) **then**
TR waits;
TH inherits priority of TR;
else
elseTR waits;
end if

[1]

2PL High Priority - pseudo code: If there is no conflict then it allowed access but if priority of Requesting is greater than holding then abort holding data item else Requesting waits for the lock.
Pseudo code:

if i **then**
f (no conflict) thenTR accesses D
else
elseif (Pri(TR)>Pri(TH)) abort TH
else
elseTR waits for the lock
end if

Priority Ceiling Protocol : Access to the data is granted to a transaction if the priority of the transaction requesting is high then the read write ceiling of all the data objects. Read Ceiling: It is the priority value of the highest transaction that writes to the database. Absolute Ceiling: The highest priority transaction reading or writing the data object. Read-Write Ceiling: It is defined dynamically at the run time.

B. Optimistic Concurrency Protocols

Optimistic Concurrency Protocols assumes less conflicts.As there are less conflicts it may results in to aborting transactions.Three phases to an optimistic concurrency control method:

Read phase: Reads data from database and stores in local variable Validation phase: Committed transactions are executed Write phase: Changes made in transaction are stored in database.

Few important optimistic concurrency control protocols: **Forward OCC:** In this protocol, transactions read and update data items freely, storing their updates into a private workplace.Before a transaction is allowed to commit, it has to pass a validation test. A transaction is aborted, if it does not pass the validation test. This guarantees the atomicity and durability properties.

OCC Broadcast Commit: In OCC Broadcast Commit (OCC-BC) protocol, when a transaction commits, it notifies to all other currently running transactions and when any conflicts

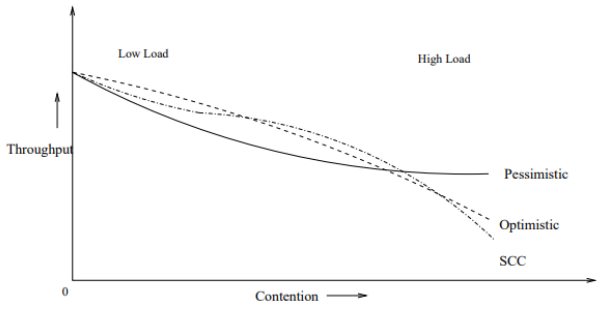

Fig. 2. Comparison of Concurrency Control Protocols

are detected, then the transaction carrying out checks it and aborts and restart the transaction.

OCC-Sacrifice: This protocol considers the priorities of transactions. It is checked that once it reaches validation stage it checks for conflicts with the currently executing transactions and if that stage problem occurs then that a transaction may be sacrificed in place of another transaction that is sacrificed later and leads to wastage of computations and deadline misses.[6]

C. Speculative Concurrency Protocol

Speculative Concurrency protocol overcomes a weakness of the OCC protocols as they check the conflicts at every read and write operations. Whenever they detect a conflict, a new version which is known as shadow version is generated of each of the conflicting transactions.As it creates version for Primary version it executes as any transaction under an OCC protocol and for Shadow version it executes under a pessimistic protocol where locking and restarts are focused.It maintains the version where there is no conflict and when conflict occurs in shadow it blocks the transaction and of primary commits then shadow is discarded.And when primary aborts, shadow works under OCC.[4]

D. Comparison of Concurrency Control Protocols

we can compare three protocols as per the figure **??** shown here.Initially all the protocols result in identical performance. When there are low conflicts OCC is the best protocol. When there is high load pessimistic protocol outperforms OCC. SCC performs better compared to both OCC and pessimistic protocols, when transactions have tight deadlines and the load is moderate.

V. Summary

The area of real-time database systems has evolved greatly over the relatively and it result of the demand to apply database technology has made it for short time of its existence.The amount of data handled by real-time systems is increasing as a structured and systematic approach to manage data.This approach combines techniques from the database systems community with the existing methods for real-time computing.This paper gives idea about transaction and scheduling of real time database.For concurrency protocol it is important to achieve transaction deadlines and certain

TABLE I
Real-time Vs General Purpose Databases

Real time Database	General Purpose Databases
Persistent data (values do not become obsolete)	Persistent + Temporal data (values become obsolete if deadline missed)
Static data items	Dynamic and Static data items
Consistent transactions maintain valid data in databases	Data validity is highly depend on the time
Logically correct and consistent	Logically correct and consistent correct Time correctness
No timing constraints	Timing constraints must be followed
Logical Consistency	Logical consistency, Temporal consistency and External consistency
To be minimized to handle more transactions	Response time requirements come from external world
Must be satisfied for every transaction	May be relaxed in few cases
Number of transactions completed per unit time	Number of transactions missing their deadlines per unit time

protocols are discussed such as 2PL which has certain drawbacks so 2PL-WP and 2PL-HP are used to overcome problems.Optimistic Protocols basically allows unrestricted data usage and works under low workload.Hence summarizing few points which are mentioned below: There exists no perfect Real time Database which satisfies all the real time database constraints. Moreover, To build a real time database it is costly as per user specifications. Limited indexing facilities are provided in real time databases
A real time database uses heavy system resources
Real time systems have a low multitasking capability
They are highly complex
Superior memory management is required in real time database systems
We need to build a task specific real time database system. A RTDB cannot be used as a generalized RTDB.

References

[1] Saud Ahmad Aldarmi. Real-time database systems: concepts and design. *REPORT-UNIVERSITY OF YORK DEPARTMENT OF COMPUTER SCIENCE YCS*, 1998.
[2] Raul Barbosa. An essay on real-time databases. *SE-412*, 96, 2007.
[3] Ben Kao and Hector Garcia-Molina. An overview of real-time database systems. In *Real Time Computing*, pages 261–282. Springer, 1994.
[4] Jan Lindström. Real time database systems. *Wiley Encyclopedia of Computer Science and Engineering*, 2008.
[5] G. Ozsoyoglu and R. T. Snodgrass. Temporal and real-time databases: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):513–532, Aug 1995.
[6] Mukesh Singhal. Issues and approaches to design of real-time database systems. *SIGMOD Rec.*, 17(1):19–33, March 1988.
[7] Sang H. Son. Research trends and issues in real-time database systems.
[8] Özgür Ulusoy. Research issues in real-time database systems: Survey paper. *Information Sciences*, 87(1):123 – 151, 1995.