An Easiest Approach To Demonstrate Process Synchronization And Concurrent Transactions

C.Bhanuprakash
Dept of Master of Computer Applications
Siddaganga Institute of Technology, Tumkur, India.

Abstract- *The intention of this paper is to approach the students in an easier ways to demonstrate the concepts of process synchronization and concurrency control of database transactions. Because, concepts of process synchronization is not so easy to deal with the students. It needs a strong evidential basics to convince them in an easier approach. Firstly, the problem we are facing here is when is the situation we face this type of problem in a computing environment. Secondly, how do we solve it in an effective way. Thirdly, why do we need to solve it ? If we don't take any measures, what would be the impact of the computational activities especially to the sharable objects/resources like main memory, printers, CPU times..etc. It presents the fundamental basics of process sychronization and concurrency control with a easier approachable ways.*

Keywords- Process synchronization, Serializability, Interleaved transactions, Concurrency control, Blind write, Lost update, Dirty read, Unrepeatable read, Locking protocols, Two – phase commit.

I. INTRODUCTION

Process Synchronization is a task of scheduling the execution of the processes in a predefined order to take care of the computer operations in a smoother ways. This is very much required for computing objects of sharable type. When a resource of a computer is shared among more than one user or processes, it leads to problem of write operations, i.e. inconsistency of data, sharable object's visibility is not clearer to any of the predefined object which leads to blind transactions, sometimes, it leads to the situation called deadlock.

In the context of operating system environment, the accessibility to sharable object needs proper synchronization mechanism. Process synchronization is a mechanism in which there should be a proper time schedule assigned between all the cooperating processes. To implement this, it needs a rule of using sharable object. That is, at a time only one process will get a chance to work with a sharable object, so that remaining co-operating processes will have to wait. They will become active only after they get a chance to work with sharable object. Preparing a time schedule to this situation needs many mechanisms like mutual exclusiveness, progress and bounded waiting. Mutual exclusiveness is a technique which allows only one process to access the critical section of other process.

Progress is another approach which tells that if no process is executing in its critical section and if other process are willing to enter in to this critical section which are not in their remainder section can decide in a fixed span of time which forces the process to enter into the critical section.

Bounded waiting is another similar approach in which processes that are willing to enter into critical section must be bounded by some fixed time slots so that a process will never be locked out of a critical section infinitely. It should be promoted immediately after it is waiting for a prolonged time.

These mechanisms will be implemented with the usage of a special kind of a integer variable of Boolean type called semaphore. It is a variable which works only in two states. i.e. True or False. It works like a electric switch i,e, ON/OFF. If there are only two processes in the competition of resource usage, then it will be used with binary semaphore which works in two states. If it is there with more than two processes, then problem will be solved by using counting semaphore. It is working like the way round robin algorithm works. Every process's usage of the sharable object will be counted in an enumerated manner so that every process will get a chance in a cycle manner. Sometimes a process gets finished its full execution during its time slot. If any process is unable to finish its execution, it will get chance in the immediate next cycle. Like this, all the cooperating process will finish their execution by accessing in to the sharable object.

Critical section is another important concept to deal with the students in a very clearer manner. It is a region in any process where the coding section will work on sharable objects. Any coding lines which works with sharable objects like global variable is a complex task. To solve this complex issue, two common approaches are used to handle critical section problems in an operating systems:

(i) Pre-emptive kernels: In this, process is allowed to be forcibly stopped or temporarily aborted while it is running in kernel mode. Because of this pre-emptive nature, these kernels should be carefully designed so as to ensure that shared kernel data are free from race conditions.
(ii) Non Pre-emptive kernels: In this, process is not allowed to be stopped / aborted while it is running in kernel mode. A process is allowed to run until it finishes its full execution. As only one process is active in the kernel at a time, a non-pre-emptive kernel is essentially free from race conditions.

In a database environment, it is common that the same data object/record is shared by more than one user at a time. These systems generally allow multiple transactions to run concurrently i.e. simultaneously. Concurrency control is the activity of coordinating many processes / users to access to a same database object / record in a multiuser environment. It permits the users/processes to access a database object in a multi user transactions environment by keeping the illusion that each of the user is executing alone on their own systems. The technical difficulty in attaining this goal is to prevent database updates performed by one user from interfering with other database retrievals and updates performed by another user/process. When the transactions are updating data object concurrently, it may lead to several problems with the inconsistencies of the data.

The commonly performed operations on any database object are read/write. If any user performing read operation on any database object, then there will be no problem, even it is accessible by many users, absolutely there will be no problem arises. Because, everybody is just performing only read operation, i.e. they just watching the current status of the database object. The same status of the object will be visible to everybody. Instead, the problem arises only when it is performing with write operations. That is updating the data object in the form of adding the new value, changing the existing value or removing an existing value. The problems that are common during concurrent transactions are dirty read, unrepeatable read, overwriting uncommitted data and lost update.

(i) Dirty read: It is a type of a transaction problem in which a transaction could read a database object that has been modified by another transaction which has not yet committed. This happens when a transaction updates a data item, but later the transaction fails. It could be due to system failure or any other operational reasons.
(ii) Unrepeatable read: If one transaction tries to read the value of a database object, it will get a different value, even though it has not been modified in the meantime. This is called an unrepeatable read. This will not happen in case of serial transactions.
(iii) Overwriting uncommitted data: This is the another problem happens when one transaction could overwrite the value of a database object, which has already been modified by another transaction, while it is still in progress. Even if first transaction does not read the value of database object written by first transaction. This problem is called overwriting uncommitted data.
(iv) Lost update: In this problem, the first transaction is about to committing the T2, by overwriting the value of the transaction T2 as set by T1 without reflect T1's update, but T1's update is lost. This is called lost update.

These problems will be solved by using locking mechanism called as lock based concurrency control. In this mechanism, it is used with a special purpose object called as lock which is nothing but a small book keeping object associated with a particular database object. To use the lock in the database environment, it is necessary to follow certain formalities. This is possible by using a locking protocols.

A locking protocol is a set of rules to be followed by each transaction to ensure that, even though actions of several transactions might be interleaved, the net effect is identical to executing all transactions in some serial order. Different locking protocols use different type of locks, such as shared locks or exclusive locks.

Most of the concurrency control algorithms fall into one of the three basic types of algorithms. i.e. locking algorithm, timestamp algorithm and optimistic / certification algorithm.

In this paper, it is trying to present a practical approach to demonstrate the concepts of process synchronization and database concurrency control to the students in an easier approaches. For this purpose, we took an example system setup where it has three users who are all willing to work on a global variable. This global variable is stored in main memory which in turn accessible to more than one user at a time. Its scope will be visible to more than one user / process at a time. If this is the case, how to perform the updates by three users simultaneously with different values is the prime concern of this paper. Because, the complexity arises here is, to decide what would be the final value of global variable after it has been updated by all three users

simultaneously. This paper deals with solutions to these kind of problems with a predefined algorithms. Similarly, it has given effective solution approaches to solve the problems related to database concurrency control.

II. RELATED WORK:

There have been many approaches made by many researchers on process synchronization as well as database concurrency control.

"Concurrency Control in Advanced Database Applications"[1]. Mr. Naser S. Barghouti and Gail E. Kaiser, made a research on concurrency control by taking in to a consideration of banking and Airline reservation system. This paper outlines the characteristics of data and operations in some advanced database applications and discussed their concurrency control requirements.

"An Efficient Approach For Concurrency Control In Distributed Database System"[2]. Md, Tabrez Quasim proposed an hybrid architecture that provide locking for high conflict data items and optimistic access for the rest of the data objects. He has tried to get rid of a distributed concurrency control performance trade-offs , by studying the performance of four related algorithms- Distributed 2PL , Wound Wait , Basic time stamp ordering and a distributed optimistic algorithm using a common performance framework. He has examined the performance of these algorithms under various degrees of contention. Data replication and workload distribution ".

The combination of these results suggest that " optimistic locking where transactions lock remote copies of data only as they enter into the commit protocol may actually be the best performer in a replicated database where messages are costly.

"Analysis of factors affecting Process Synchronization"[3] Mrs Aarthi Chhikara proposed various methods to ensure the orderly execution of the cooperating processes. She also discussed several hardware based problems along with solutions that ensures mutual exclusion. She also disclosed a mutex locks to solve a critical section problem. He also highlighted the importance of several classical synchronization problems such as producer-consumer problem, readers-writers problem, dining philosophers problem.

"An Approach to Process Management using Process Synchronization"[4], Deepti Sindhu, Anupma Sangwan and Kulbir Singh focussed mainly on how to manage the multiple processes on multi-processors environment. Since, earlier version of process management dealing with synchronizing the processes with an uni-processor system where as present days systems are dealing with multiprocessor's system. Ultimately, a Processor can be made more productive by any type of operating system if a processor switches properly between processes. It is possible if a processor is synchronized by proper synchronization of processes.

"Process Synchronization" [5], K.C.Wang, It explains the concepts of concurrent processes, the basic principle of process synchronization and the hierarchical relations among the synchronizing tools. It elaborates on implement of critical regions and use them in process synchronizing tools which include sleep/wakeup for uni-processor systems and semaphores for multiprocessor systems. It covers the applications of semaphores in concurrent programming environment. It explains situation of deadlock, its handling and shown how to devise concurrent algorithms to avoid deadlocks. It also explains the concepts and usage of pipes and applies to the producer-consumer problem to implement pipes in the MTX kernel.

III. METHODOLOGY

3.1 Experimental Setup to demonstrate Process Synchronization:

Consider a global variable 'X' which presently stored in a main memory of a computer and is currently sharing with three users, User-1, User-2 and User-3 respectively. The initial value of 'X' is zero. The user-1 is willing to work on this variable 'X' by updating its value by 10. The user-2 is willing to work on this variable 'X' by updating its value by 20. Similarly, the user-3 is also willing to work on this same variable and updating its value by 30. All the three users are trying to work on this variable simultaneously. If this is the case, what would be the final value of the variable 'X' ?. This system setup is shown in the figure – 1.

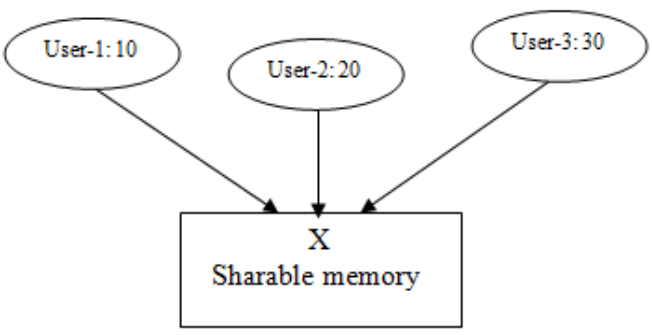

Figure - 1

To say the answer to this question is not so easy, because, we cannot say the final value of X is either 10, 20 or 30. Sometimes, the students will say that the final value of 'X' is 60 by summing all the values. i.e. 10+20+30 = 60. Even then, this is also not correct. In other occasions, the students will say the answer so smartly that the value of the variable 'X' is depends on the user who has updated the variable at last. But, we don't know which user updated at last. This situation leads to ambiguity to the teachers many times. In order to give the correct answer to this question, we need to convince the students with the concept of process synchronization. That is, making a time schedule to all the three users to work on this global variable 'X' with a principle of mutual exclusion. In this mutual exclusion, if a user-1 is working on the variable 'X', the other users, user-2 and user-3 have to wait till the updating of user-1 on 'X' has to finish. During this time, the variable 'X' should be protected for any kind of write operations from other users except the user-1. He can make any kind of write operations (i.e. adding new value to the variable 'X', changing the existing value of the variable 'X' or removing the existing value of the variable 'X'). When once the updating operation is finished, the write protection constraint applied on the variable 'X' will be released. When this is done, it is the time to apply the chance of working with the variable 'X' to any users. i.e. either user-2, either-3 or even user-1 also. To decide which user will get a chance to work with this variable depends on the algorithm being used to solve this problem. To solve this problem, we have many algorithms like First Come First Serve, Priority based algorithm, Shortest job first algorithm, Least recently used algorithm, Round Robin algorithm …etc. This is the common strategy being used for all the objects of sharable types. The same concept in database environment is called as concurrent transactions or simultaneous transactions. The techniques used to solve this problem is a called as concurrency control.

In general situations, this strategy is implemented by using special type of a variable called semaphore. Semaphore is a integer variable which works on two states. That is Wait and Signal which is similar to Boolean type of variable which works on the concept of True and False. It is also called as a locking variable which facilitate only one process to share the variable at a time and keeps the remaining process in a waiting status.

The working of wait and signal operations are:

```
wait(S) : while S <= 0
do skip;
S := S - 1
Signal(S): S := S + 1
```

Counting semaphore values will span from 1,2….N to unrestricted limit. They will be used to accessibility control to a defined resource which contains fixed number of instances. Assume, if there are 6 resources in a sharable environment, then the variable semaphore is initialized with value 6. Any of the process willing to use a resource performs an operation with wait( ) signal on the variable semaphore by decreasing the counting value i.e. 6 – 1 = 5. When an updating operation is completed, the acquired resource is released by a process by changing the status of the semaphore as signal( ) by increasing the counting value i.e. 5 + 1 = 6. When the value of a variable semaphore becomes zero, it indicates that all resources are being used/busy. Any other process willing to use a resource will getting blocked until the counting value becomes greater than zero. i.e a process releases a resource.

Binary semaphores are generally termed as mutex locks that provide mutual exclusion.

Let us assume two processes: P1 with a statement-S1 and process: P2 with a statement S2 running concurrently. Assume that statement-S2 be executed only after the statement S1 has finished its execution. This logic can be implemented readily by allowing P1 and P2 share a common variable **S**, which is initialized to 0.

In process P1, we insert the following statements

```
S1;
Signal(S);
```

In process P2, we insert the following statements

```
Wait(S);
S2;
```

Execution flow will be

1. S1 starts execution
2. Signalling the variable semaphore
3. The waiting process P2 receives the signal
4. S2 starts execution by keeping S1 in waiting state.

Implementation Semaphore: Implementation of this semaphore with following condition as shown in following code:

```
struct semaphore
{
initialvalue;
Lists *L ; // process
lists
}
```

Here each semaphore is having an integer value and a list of processes L. The waiting processes on semaphore are added to the list of processes. Signal operation removes one process from process list and wakes up that process.

The semaphore operations can be now written as follows:

Syntactical description of Wait Operation:

```
Wait(S) // wait
operation
{
S.val--;
If (S.val < 0)
{
add this process to S.L;
block;
}
}
```

Syntactical description of Signal Operation:

```
Signal(S) //signal
operation
{
S.val++;
If (S.val <= 0)
{
remove a process P from S.L;
wakeup (P);
}
}
```

Here, the block operation suspends the process which invokes it and it resumes execution of blocked process by the wakeup () operation. The operating system treats these two basic operations as basic system calls.

3.2 Concurrency Control in Database Applications:

It is considered to be one of the database management activities of co-ordinating the multiple actions of database manipulating processes that separates concurrency that access shared data and can effectively ensures the isolation with one another. The main role of concurrency control is to ensure the serializability of concurrently executing transactions.

During the concurrent transactions, read and write operations performed on sharable data object may cause conflict among these transactions. To deal with these conflicts a concurrency protocol has been devised which will work with two basic stages.

i) It is trying to avoid conflicts or else it is trying to detect conflicts.
ii) When once they detected it will eliminate them.

There are many methods to allow concurrent transactions by make use of locking mechanism, Lock is nothing but a book keeping object kept on the database object being updated during concurrent transaction. It is a mechanism in which we put lock on the required database record or group of records. When once it is locked, no other process can make any changes to the selected records. Only authorized process can do any changes it wants. Lock will be released automatically when the transaction is completed. This is possible by make use of many methods and protocols. Following are the popular methods being used.

i) Two phase Locking : In this method, Locking can be implemented in two phases. In first phase, it is selecting required records by using conventional SQL queries and apply the lock. Following is the SQL query with a syntax:

```
SQL> Select <List of Columns/attributes>
from <Table_Name>
Where <condition>  FOR UPDATE.
```

In the second phase, It will perform the required updating operations and complete the transactions by ending with COMMIT statement. Following SQL syntax is used.

SQL> <Update / Delete database object_name from <Table_Name> WHERE CURRENT OF Cursor_name.

Where cursor is the SQL variable which keeps the result set of ongoing select statement. When once it is given with COMMIT statement, the transaction is completed and which automatically releases the already applied locks. Now, this record is ready to update from other users / processes. This will continue throughout the database session.

ii) Wound Wait (WW) method: This is the second method which works on the concept of a slogan “Get ready any and Write all”. More or less, it is trying to avoid deadlock rather than reducing waiting time of remaining processes. Here, deadlock will be avoided with use of timestamps. Each transaction is numbered so that initial process will get a value 1 and last transaction will get a value ‘n’.

iii) Time Stamp Ordering: It works based on the principle of new and old data items. That is, a transaction's operation performed on a data item is executed only if its timestamp is newer than the timestamp of all transactions that have previously accessed the data item. The scheduler generated by

timestamp methods are serializable. Timestamp must be drawn from a totally ordered domain – this is usually achieved by using tuples composed of time value as primary ordering field and the transaction manager's unique number as secondary field .

iv) Distributed Certification: This is the fourth method works on the basis of distributed timestamp on Optimistic Concurrency Control algorithm. It operates by exchanging certification information during committing protocol. For each of the data item, a read timestamp and a write timestamp are maintained. Transactions may read and updates data items freely , Storing any updates into a local workspace until commit time. For each read, the transaction must remember the version identifier associated with the item when it was read, then when all of the transaction's cohorts have completed their work and have reported back to the master, the transaction is assigned a globally unique timestamp.

IV. CONCLUSION

This is the small attempt put to reduce the efforts in convincing the students to teach the topics on process synchronization as well as concurrency control in an easier ways. On many occasions, a common complaint we are getting in the subject Operating systems that, teachers may telling the students to teach this module after the completion of remaining modules or else at the end of the semester. But, most of the occasions, they may skip this module. Because, they don't have minimum confidence to deliver the contents of the chapter. Even students are not willing to listen this so effectively, because of its complex nature of concepts. By observing all these situations over a span of many years, I thought of presenting these simple approaches to you in a form of a paper. Now also, I was not able to cover all the concepts of process synchronization and concurrency control, but I have covered the major portion of the module which seems to be complex one. I made my sincere attempt in this regard to deliver the contents shown in this chapter. I would highly appreciate your valuable feedback, healthy comments and suggestions.

REFERENCES

[1] NASER S. BARGHOUTI AND GAIL E. KAISER : "Concurrency Control in Advanced Database Applications" – Department of Computer Science, Columbia University, Newyork.
[2] Mohammed Tabrez Quasim "An efficient Approach for Concurrency Control In Distributed Database System" – Researchgate, Article published in January 2013.
[3] Aarthi Chhikara "Analysis of Factors Affecting Process Synchronization" – Journal: International Journal of Computer Science Engineering (IJSCE), Volume – 9, No-02, Issue – 4, ISSN 0975 – 3397. Published online February 2017.
[4] Deepti Sindhu, Anupma Sangwan, Kulbir Singh, "An Approach to Process Management using ProcessSynchronization" – International Journal of Computer Applications, Volume-128, Number – 7, Year of Publication – 2015.
[5] K.C. Wang, "Process Synchronization" Book Chapter-5, DOI: 10.1007/978-3-319-17575-1_6, June - 2015
Book Title : Design and Implemenation of MTX Operating System

AUTHORS PROFILE

C.Bhanuprakash, received the B.E.in Mechanical Engineering and Master of Computer Applications degrees from Siddaganga Institute of Technology, Tumkur, of Bangalore University, in 1993 and 1998, respectively.

C.Bhanuprakash has 19 years of experience in Teaching, 4 Years of Industry experience and 8 years of research experience. Presently he is working as Assistant Professor in the department of Master of Computer Applications, Siddaganga Institute of Technology, Tumkur, Karnataka. Currently, he is pursuing his Doctor of Philosophy at Research Centre, Department of Computer Science and Engineering, SIT, Tumkur of Visveshvaraiah Technological University, Belgaum, in the field of soft computing techniques, database and data mining applications.

His areas of specialization include design and development of database applications, Operation Research, Optimization Techniques, Intelligent Data Analysis, Advanced Operating systems, Neural Networks, Data Mining and other Soft Computing Techniques.