International Journal of Electronics Communication and Computer Engineering
Volume 2, Issue 2, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

Algorithms of Buffer Management in Active Real-Time Database System

Mrs. Pranjal S. Bogawar
Assistant Professor
Priyadarshini College of Engineering., Nagpur University, India
pbogawar@yahoo.com

Abstract **- Active real-time databases have emerged as a research area in which concepts of active databases and real-time databases are combined into real-time database with reactive behavior. Such systems are envisioned as control systems for environments as diverse as process control, network management and automated financial trading. Sensors distributed throughout the system report the state of the system to the database. Unacceptable state reports typically results in corrective actions being triggered with deadlines. This paper is a study of various buffer management techniques in real time database management system & active real time database management system as well as we implemented few of them. We created ARPLRU (Active Real-Time Priority LRU) and ARPHA (Active Real-Time priority hint algorithm) and compared with PAPER algorithm.**

I. INTRODUCTION

As the amount of information in a system grows larger, it needs to be stored somewhere and stored intelligently, with regard to space requirement and time for retrieval. Applications that include a database system work in real time environment, traditional database is not supportable. Hence the evolution of real time database system and active real time database system. In real time database system we are have timing constraints. In active real time databases we are considered both timing constraints as well as actions are being triggered.

Buffer management is associated with the use of main memory during reading and writing data from and to the disk. In the system when main memory is limited, high priority transactions are executed first. When main memory has plenty of space, database is preferred to reside in the main memory. But when the system crashes total data in main memory is lost. Hence buffer management techniques are very much preferred. Here we are buffer considering buffer management techniques used in both real time as well as real time active database system.

II ACTIVE DATABASE

An *active database* is a database that includes active rules, mostly in the form of ECA rules[4].

A. EVENT CONDITION ACTION (ECA)

ECA is active rules in event driven architecture and database systems.

Such a rule did traditionally consist of three parts:

- The *event* part specifies the signal that triggers the invocation of the rule
- The *condition* part is a logical test that, if satisfied or evaluates to true, causes the action to be carried out
- The *action* part consists of updates or invocations on the local data

III . REAL TIME DATABASES

A real-time database is a processing system designed to handle workloads whose state is constantly changing. This differs from traditional databases containing persistent data, mostly unaffected by time. Real-time processing means that a transaction is processed fast enough for the result to come back and be acted on right away. Real-time databases are useful for accounting, banking, law, medical records, multi-media, process control, reservation systems, and scientific data analysis[6].

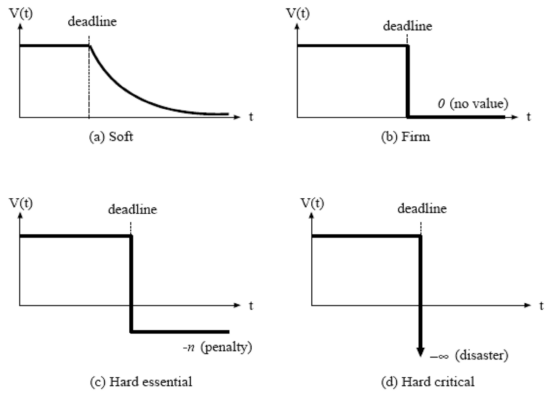

Fig 1

A system that correctly perceives the serialization and timing constraints associated with transactions with soft or firm deadlines, takes advantage of absolute consistency.

Hard deadline: If not meeting deadlines creates problems, a hard deadline is best. It is periodic, meaning that it enters the database on a regular rhythmic pattern.

Firm deadline: Firm deadlines appear to be similar to hard deadlines yet they differ from hard deadlines because firm deadlines measure how important it is to complete the transaction at some point after the transaction arrives. Sometimes completing a transaction after its deadline has expired may be harmful or not helpful, and both the firm and hard deadlines consider this. An example of a firm deadline is an autopilot system.

Copyright © 2011 IJECCE, All right reserved

Soft deadline: If meeting time constrains is desirable but missing deadlines do not cause serious damage, a soft deadline may be best.

IV. ACTIVE REAL TIME DATABASES

Active Real-time databases are a combination of active database system with time constraints in the form of deadlines that must be met. Predictability is the paramount importance in real time databases and reactive mechanism add unpredictability.

There are periodic transactions as well as aperiodic transactions. In periodic transaction most of the times the sensor sense the data periodically and updates the database. In aperiodic transaction user trigger the action and there is the updations or changes are done in the databases.

When a rule is triggered the execution time of the transaction in which the action is executed will increase. This means that the slack, the time between the estimated end of transaction and the deadline, will decrease. In extreme cases the slack will become negative and the deadline will be missed. In extreme cases the slack will be negative and deadline will be missed. Every rule action should be run in detached mode to manage the execution time. This does not increase the transaction time but increase the load on the system, if the transactions run concurrently.

An alternative is to actively take the extended execution time into account when triggering action and rescheduling transaction needed. If the legal schedule cannot be obtained, the action could be rejected, the transaction could be aborted, or contingency plan(Some part of the task is skipped/ task is spited into multiple small tasks/ several versions can be implemented) could be executed[3,9].

V. BUFFER MANAGEMENT

A *Buffer Manager*

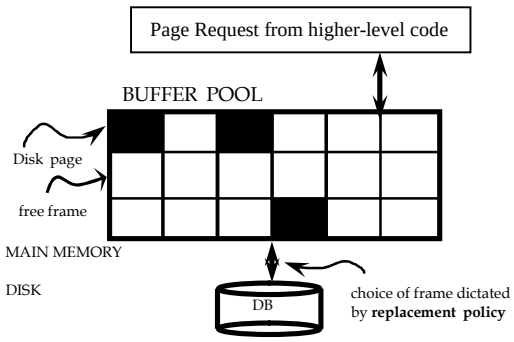

Fig 2: Buffer Manager[5]

The buffer manager is a software layer that is responsible for bringing pages from disk to main memory as needed. The buffer manager manages the available main memory by partitioning it into a collection of pages, which we collectively call buffer pool. The main memory pages in buffer pool are called as frames.

B *Buffer Model*

[8]Buffer organization consists of two types of buffers- a pool of private buffers and a shared global buffer.

Private Buffers are used as transaction working space for the purpose of database recovery and used for only personal use of particular transaction.

The global buffer management consists of two processing components:

- *Buffer Allocation* - which distributes global buffer space among concurrent transactions.
- *Buffer replacement* - which is responsible for accessing of the global buffer and page replacement operations.

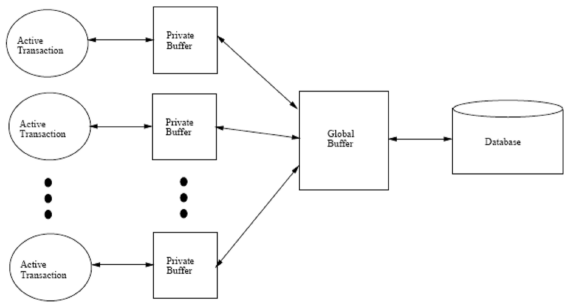

Fig3:Buffer Model [3,8]

Upon arrival of new transaction, buffer allocation component decides if , and how much, global buffer space can be allocated to the transaction, according to a certain buffer policy. When the transaction makes a request for a page, it will first access its private buffer. If the requested page is not found, buffer replacement component searches the global buffer space allocated to it. If there is a hit in the global buffer, transaction will use the page copying it to its working space. Otherwise, the buffer replacement component will fetch the requested page from disk to global buffer, where the page replacement operations takes place if there is no free space available in the buffer. Consistency between database and global buffer must be maintained.

C. Buffer Allocation

The function of buffer allocation is to distribute the available buffer space among concurrent transactions. Buffer allocation appears more important when there is contention for the global buffer. The basic idea for buffer allocation is to allocate the global buffer space to the real-time transaction in such a way that the overall transaction deadline guarantee ratio can be improved compared to buffer allocation scheme which ignores the timing information. It considers following policies for buffer allocation.

- *Alloc0* – Allocating buffer to all the concurrent transactions. Under this, all the transactions are treated equally in using global buffer regardless of timing constraints.

- *Alloc1*- Allocating global buffer to the transaction which has earliest deadline among the concurrent transactions.

Copyright © 2011 IJECCE, All right reserved

- *Alloc2*- Let Ti(i=1,2,….n) be the total of n concurrent transactions in the system. The allocation scheme is described by following algorithm.
  1. Sort Ti by Transaction deadline in ascending order
  2. allocate the global buffer to the first m Tj's such that the following condition holds.
  3.

$$\sum_{j=1}^{m} Tj.ws < G\_Buffer\_size < \sum_{j=1}^{m+1} Tj.ws \text{ --(1)}$$

Where Ti . ws – the working set of transaction Ti

- *Alloc 3*- Allocating the global buffer to any m concurrent transaction such that eq.1 holds. This policy does not address transaction timing constraints. This is used for comparing the results of allocation policy alloc2.

The transaction is called buffer owner if it has been allocated the global buffer space by any buffer allocation policy used; otherwise it is called a buffer user. Whenever page references are not in private buffer, whether it is buffer owner or user it will search first in global buffer. If there is buffer hit it will transfer that page from global buffer to private buffer. But when there is buffer miss, buffer owner fetch the missing page from disk to the global buffer, while buffer user only read the missing page from the disk to its private buffer[8].

D. Buffer Replacement

Comparative study of Different Algorithms

Chara-cteristics / Algorithms	Admission Policy	Replacement Policy	Priority	Anticipatory Fetch	Database Implementation
LRU	Depends upon the Buffer size	LRU	NO	No	Traditional
LRU-K	Depends upon the Buffer size	Pages whose backward distance is maximum of all pages in buffer	No	No	Traditional
LRU-dl	Depends upon the Buffer size	Pages of transactions which are more time ctritical can't be easily replcaed	No	No	RTDB
Priority Hints	Depends upon the Buffer size and Priority	Favoured pages by MRU and normal pages by LRU (Favoured pages of a Transaction can be stolen only by the transactions of higher priorities.)	Yes	No	Traditional
Priority DBMIN	Depends upon the Buffer size and Priority	Depending upon query, policy changes. It can be LRU,MRU,LIFO etc.	Yes	No	Traditional
PAPER	Depends upon the Buffer size and Priority	Clean pool pages will be vicitmized first by FCFS otherwise pages of dirty pool are used by FCFS policy. But in Reuse pool pages are sorted according to deadline and clean-bit. And higher key pages are replaced first.	Yes	Yes	ARTDB
Tree-Based	Depends on the Buffer size	Level control strategy is used.	No	No	RTDB

Table 1: Comparative study of algorithms

The replacement policy comes into play if there are no free buffer frames for newly fetched frame. So there are various algorithms to tackle this problem. The comparative study of all these algorithms are given in above chart[9,14,15,16,20].

V IMPLEMENTATION

A. Simulation Model for Buffer Management in Active Real-time Database System

We created the simulation model for the implementation of buffer management in Active Real-time Database System in Java and SQL server. For storing the results we created various tables. Also some assumptions are made.

Assumptions are

- One query require a one page.
- One Page is of size 1MB.
- The Global Buffer of size 12 MB. Give the buffer size as a multiple of 3.
- Every Transaction contains the sub-queries.
- Priority of transaction should be 1,2,3.
- Deadline is user input.

Inputs

Transaction id
Transaction (Set of queries)
Deadline
Priority
Buffer Size
Arrival Time
Number of transactions to be executed.

Outputs

Completion time= Arrival time + deadline
Page Faults – If query is not in buffer
Page Hits – If query is in buffer
Successful Transaction
Transaction Miss Ratio

$$Transaction\,Miss\,Ratio = \frac{Number\,of\,transactions\,that\,aborts * 100}{Total\,number\,of\,transactions}$$

Page Fault Rate

$$Page\,fault\,rate = \frac{Number\,of\,page\,faults * 100}{Number\,of\,page\,requests}$$

Tables

We created various tables to store the various results.

Transaction Table

Transaction table stores the various transactions. This table is used to select the number of transactions to be executed.

Various fields of the transaction tables are as follows.

Id : This is the numeric field, which contains the number of the transaction.
Query : This field stores the group of queries.
Priority : Stores the Priority of the Transaction.
Deadline: Stores the deadline i.e. the duration in which the query should be completed successfully.

Copyright © 2011 IJECCE, All right reserved

International Journal of Electronics Communication and Computer Engineering
Volume 2, Issue 2, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

Trigger Table

This table stores all the triggers fired on to the table. Various fields of the table are as follows

Triggerid : This assigns the particular id to the trigger

Table Name : This field is used to specify on which table you have to fire the trigger

Command : This field specifies on which command e.g. insert ,update or delete we have to fire the trigger

Triggerquery: This field stores the query which we have to fire.

Result Table

This table stores the end result of all the algorithms. Various fields of this table are as follows.

Id : This field gives the particular unique representation for the row.

Algoid : This stores the particular id for the particular algorithm eg. For ARPLRU 1, ARPHA 2, and for PAPER 3.

Algoname : It stores the name of the algorithm

Transaction : This stores the how many transaction get executed

Failure : Out of total queries how many are not into the buffer.

Hit :Out of total queries how many are already in the buffer.

Buffer: It specifies the buffer size for that particular transaction.

Miss : Specifies how many transactions get failed.

Algorithms

We designed the two algorithms ARPLRU and ARPHA to compare their results with PAPER algorithm.

ARPLRU (Active Real-time Priority LRU)

1. Transactions are coming according to arrival rate (Transactions per second).
2. Place the transactions in the priority Buffer Pool according to their priority.
3. [set Arrival Time, End Time]
   a) For the transaction set Arrival Time
   b) For all transactions set the following parameters
   set End Time= Arrival Time + deadline
4. Rule manager is called for every query of the transaction which checks what will be the sub-transactions fired and prefetched into prefetch pool and are serialized after the main transaction execution.
5. In the priority buffer pool the transaction whose deadline is most time critical is executed first.
   Test conditions for Buffer Allocation and Replacement:
   I. It first checks whether transaction pages are already present in the buffer are not.
   If already in buffer then
   Hit=Hit + 1
   Else
   checks buffer space is available for the newly arrived transaction or not.
   a. If the buffer space is available for this transaction then buffer space is allocated to this newly arrived transaction according to their priority level.
   b. If the buffer space is not available then it checks from the lowest priority Buffer Pool whether deadline of any transaction is expire or not.
   c. Page fault = Page fault + 1
   II. If deadline expired then that transaction is replaced by the newly arrived transaction and page fault is incremented by one. And if not LRU policy is used for the replacement and again page fault is incremented by 1.
   III. If all the queries of the transactions are executed properly then transaction success is increased by one.
   Step 2 to step 5 is repeated for every transaction.

ARPHA (Active Real-time Priority Hints Algorithm)

1. Transactions are coming according to arrival rate (Transactions per second).
2. Place the transactions in the prioterized Buffer Pool.
3. [set Arrival Time, End Time]
   a) For the transaction set Arrival Time
   b) For all transactions set the following parameters
   set End Time= Arrival Time + deadline
4. Rule manager is called for every query of the transaction which checks what will be the sub-transactions fired and prefetched into prefetch pool and are serialized after the main transaction execution.
5. In the prioterized buffer pool the transaction whose deadline is most time critical is executed first.
6. In this algorithm, there are two types of pages favored and normal. Again favored pages are of two types “Fixed” and “Unfixed”. Transactions which are not executed and whose deadlines are not expiring are “Fixed” transactions. And if transactions are executed within deadline are “Unfixed”.
   Test conditions for Buffer Allocation and Replacement:
   I. It first checks whether transaction is already present in the buffer pool are not.
   II. If it is then Hit is incremented by one otherwise page fault is incremented by one.
   III. And if not then it checks buffer space is available for the newly arrived transaction or not.
   IV. If the buffer space is available for this transaction then buffer space is allocated to this newly arrived transaction according to its priority.

Copyright © 2011 IJECCE, All right reserved

International Journal of Electronics Communication and Computer Engineering
Volume 2, Issue 2, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

V. If the buffer space is not available then it checks from the lowest priority Buffer Pool whether deadline of any transaction is expire or not i.e. means it checks whether there is an "Unfixed" transaction are not. If there is no unfixed in lowest priority buffer then it will check for middle level priority transaction. Even though there is no unfixed pages in the middle priority transaction then it will check for unfixed page at Higher priority transaction.

VI. If it is then that "Unfixed" transaction is replace by the newly arrived transaction using global MRU policy and page fault is incremented by one. And if not LRU policy is used for the replacement of "Fixed" transaction and again Failure is incremented by one.

Step 2 to Step 6 is checked for all transaction.

Observations:

First observation is the graph which is drawn across the successful transaction to the arrival rate(Transactions per second fired). In ARPLRU the successful transaction rate is lower than ARPHA and PAPER. Hence PAPER is very efficient algorithm.

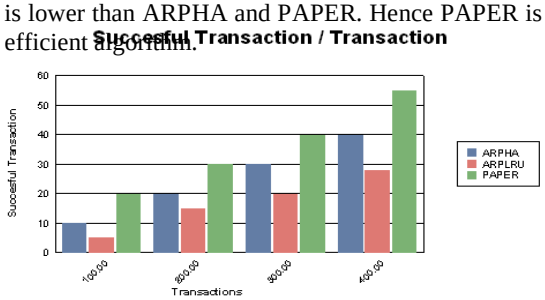

Fig4: Successful Transactions/Transactions

Second observation is the graph, which is drawn across the Transaction miss ratio and arrival rate per second. Fig 5 shows that the PAPER algorithm is very good. The efficiency of ARPHA is very much near to PAPER algorithm.

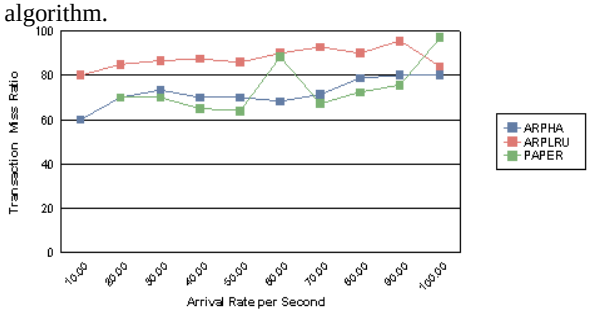

Fig 5: Comparison of Buffer Management algorithms for Transaction Miss Ratio versus Arrival Rate

Third observation is the graph, which is drawn across Page fault rate and Transactions per second as shown in fig.6.

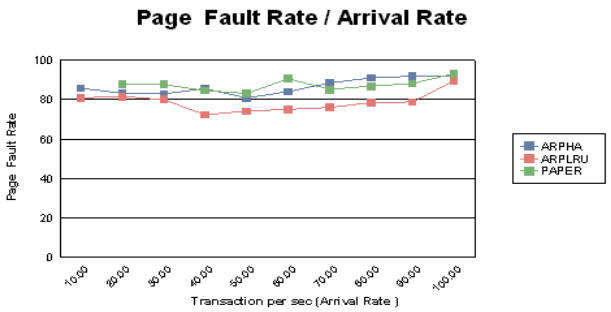

Fig 6:Compararision of Buffer management algorithms for Page Fault Ratio versus Arrival Rate

VI CONCLUSION

In this paper we have studied various techniques used for buffer management in real time databases and active real time databases. In buffer management mainly buffer allocation and buffer replacement. Buffer allocation is used when buffer is free. Otherwise buffer replacement policy is used.

The Active Real-time database model on traditional database system has been simulated here. Three buffer management algorithms are investigated and implemented in Active Real-time Database System. These are ARPLRU(Active Real-time Priority Least Recently Used), ARPHA (Active Real-time Priority Hints algorithm), PAPER(Prefetching Anticipatorily and Priority basEd Replacement). These algorithms are compared with each other. All the algorithms consider the active transaction as well as Real-time parameters like priority and deadline. These algorithms execute the sub-transaction in detached mode. The sub-transaction are fetched in prefetch pool and are serialized after the main transaction execution. Because of this deadline of main transaction did not missed and the transaction miss ratio is minimized. These algorithms also consider the page hit and page fault of the sub-transactions.

The result of comparisons shows that PAPER is very efficient than the ARPHA and ARPLRU. For most of the workloads ARPHA is very much efficient than the ARPLRU. ARPHA performs almost near to the PAPER. From transaction miss ratio graph is it shown that the transaction miss ratio will increase as the number of transaction will increase.

REFERENCES

[1] Bhaskar Purimetla, Rajendran M Shivshankaran, John A. Stankovic, Krithi Ramamrithm & Don Towsley. "Priority Assignment in RealTime Active Databases". Department of Computer Science, University of Massachusetts, Amherst, MA 01003, citeseer.ist.psu.edu/sivasankaran94**priority**.html - 25k, 0-8186-6400-2/94, 1994 IEEE

[2] Uwe Brinkschulte. "Tree Based Buffer management in Real-Time Database Systems". Institute of Microcomputers and Automation, University of Karlsruhe Haid-und-Neu-Str. 7,76131 karlsruhe, Germany, citeseer.ist.psu.edu/510842.html - 21k

[3] Anindya Datta, Sarit Mukherjee, Igor R. Viguier. " Buffer Management in Active, Real- Time Database Systems". January 21, 1996, citeseer.ist.psu.edu/110224.html - 31k,

[4] Mohamad Ridha. " Active Database Implementation For Real-ime Computing", Lisensi Dokumen, Copyright © 2005 IlmuKomputer.com

[5] Database management system by Raghu Ramakrishnan,

Copyright © 2011 IJECCE, All right reserved

Johannes Gehrke.
[6] Jokim Eriksson. " Real-Time and Active Databases : A Survey". Department of Computer Science, University Skovde, December 17,1996.
[7] Umeshwar Dayal, Eric N. Hanson, Jennifer Widom. "Active Database Systems" IN: Won Kim,editor, Modern Database Systems: The Object Model, Interoperability, and beyond, Addision-Wesley,Reading, Massachusetts, Sept.1994
[8] Jiandong Huang and A. Stankovic. " Buffer management in Real- Time Databases", Department of Electrical and Computer engineering, Department of Computer and Information Science, university of Massachusetts, Amherst, MA 01003
[9] Anindya Datta, Sarit Mukherjee, Igor R. Viguier. " Buffer Management in Real- Time Active Database Systems", IEEE transactions on systems, main and cybernetics-Part A : Systems and Humans, Vol 29 NO.2, March 1999
[10] Sang H. Son, Carmen C. Iannacone and Marc S. Poris, " RTDB: A Real-Time Database manager for Time- Critical Applications", Department of Computer Science, University of Virginia, Charlottesville, Virginia 22903, USA. 0-8186-2212-1/91/0000/0207$01.00 © 1991 IEEE
[11] Nandit Soparkar, Henry Korth, Abraham Silberschatz. "Databases with Deadline and Contigency Constraints"
[12] P.A. Fishwick. Simpack: Getting started with simulation programming in C and C++. Technical report TR92-022, Computer and information Sciences, University of Florida, Gainesville, Florida 1992.
[13] Hong-Tai Chou, David J.DeWill, "An Evaluation of Buffer management Strategies for Relational Database Systems", Computer Sciences Department University of Wisconsin, Proceedings of VLDB85, Stockholm.
[14] Elizabeth J O'Neil, Patrick E. O'Neil, Gerhard Weikum, " The LRU-K page Replacement Algorithm for database Disk Buffering",Department of Mathemetics and Computer Science University of Massachussetts at Boston and ETH Zurich Switzerland, SIGMOID/5/93/Washington,dc,usa©1993 acm 0-897915925/93/0005/0297…$1.50
[15] Rajiv Jauhari, Michael J.Carey,and Miron Livny, "Priority-Hints:An Algorithm for Priority-Based Buffer Management", Computer Sciences Department University of Wisconsin, Madison, WI53706
[16] Andrew S. Tanenbaum, "Modern Operating Systems"
[17] Henry Korth, "Database system Concepts".
[18] Carlo Zaniolo, "Active Database Rules with Transaction Conscious Stable Model Semantics" Computer Science Department, University of California, Los Angeles, CA90024.
[19] Klaus R. Dittrich, Stella Gatziu, Andreas Geppert, " The Active Database Management System Manifesto: A Rulebase of ADBMS Features" Proc. 2nd Workshop on Rules in Databases (RIDS), Athens, Greece, September 1995. Lecture notes in Computer Science, Springer 1995.
[20] Theodore Johnson, Dennis Shasha, " 2Q: A Low Overhead High Performance Buffer management Replacement Algorithm", Proceedings of 20th VLDB Conference Santiago, Chile, 1994, pp. 439-449.
[21] Gultekin Ozsoyoglu, Richard T. Snodgrass, "Temporal and Real-time Databases: A Survey", IEEE Transaction on Knowledge and Data Engineering, Volume 7, Issue 4,August 1995
[22] Stella Gatziu, Klaus R. Dittrich, "SAMOS: An Active Object-Oriented Database System", IEEE Bulletin of the TC on Data Engineering, January 1993.

Copyright © 2011 IJECCE, All right reserved