A PETRI NET MODEL FOR AN ACTIVE DATABASE SIMULATOR

Joselito Medina-Marín[(a)], Xiaoou Li[(b)], José Ramón Corona-Armenta[(c)], Marco Antonio Montufar-Benítez[(d)], Oscar Montaño-Arango[(e)], Aurora Pérez-Rojas[(f)]

[(a) (c) (d) (e) (f)]Autonomous University of Hidalgo State, Advanced Research in Industrial Engineering Center, Carr. Pachuca-Tulancingo Km 4.5 Col. Carboneras, Pachuca, Hidalgo, México
[(b)]Research and Advanced Studies Center of the National Polytechnic Institute, Av. Instituto Politécnico Nacional No. 2508, Col. Zacatenco, México, D.F., México

[(a)]jmedina@uaeh.edu.mx, [(b)] lixo@cs.cinvestav.mx, [(c)]jrcorarm@yahoo.com, [(d)]montufar@uaeh.edu.mx, [(e)]oscarma11@hotmail.com, [(f)]auropr@yahoo.com,

ABSTRACT

Active database systems were introduced to extend the database functionality. As well as a repository of data, active database can detect the occurrence of events in a database system and react automatically to event occurrence and execute certain actions either inside or outside the database. This behavior is specified by means of ECA (event-condition-action) rules, i.e., when an event has occurred, if the condition is evaluated to true, then an action is executed. In this paper a simulator for active databases, named ECAPNSim, is described. ECAPNSim uses the definition of ECA rules like a structure of an extended Petri net model, the Conditional Colored Petri Net (CCPN). Conditional Colored Petri Net definition involves the knowledge and execution model, which describe the features that an active database system must have. An example has been developed in order to show the ECAPNSim applicability in a certain study area.

Keywords: Petri nets, active database, ECA rule, simulation.

1. INTRODUCTION

Traditional databases (DB) were developed to store a huge amount of information. In this DB type the information only is accessed by insert, delete, update and query algorithms, which were previously programmed in a Data Manipulation Language (DML) by the DB administrator. The set of all this data manipulation programs is the Database Management System (DBMS). However, the execution of those programs is performed only by the request of either a DB user or the DB administrator.

Nevertheless, there are systems that cannot be implemented by using a traditional DB approach. Such systems are those where is well known that if certain events occur in the DB and if the DB state satisfies certain conditions, then an action or procedure is performed in the DB. Therefore, it is necessary to use an approach where a DB could have the ability to react automatically when an event occur either inside or outside DB environment, after this, it can verify the DB state to evaluate conditions, and if condition is evaluated to true it can execute procedures that modify the DB state. In order to provide of active behavior to traditional DB, Active Databases (ADBs) were introduced. If a human being takes charge to detect the event occurrences, verify conditions, and execute procedures instead an ADB system, then the system may not work well. Thus, it is very important to add enough information to DB about the active behavior and convert a traditional DB into an Active one.

Active behavior of a DB can be defined through a base of active rules, which has the specification of events that will be detected, conditions that will be evaluated, and actions or procedures that will be performed in the DB. The model most widely used is the event-condition-action rule (ECA rule) model, whose general form is as follows (Silberschatz, Korth, and Sudarshan 1999):

on event e_1
if condition c_1
then action a_1

ECA rule model works in the following way: when an event e1 that modifies the current DB state occurs, if condition c1 is evaluated to true against DB state, then either an action a1 is executed inside DB or a message is sent outside DB.

An event e1, which can trigger to an ECA rule, can be of two types: primitive event or composite event (Paton and Diaz 1999). A primitive event is generated by the execution of an operation over the DB information (insert, delete, update, or select), a DB transaction, a clock event (which can be absolute, relative, or periodic), or the occurrence of a DB external event. On the other hand, composite events (disjunction, conjunction, sequence, closure, times, negation, last, simultaneous, and any) are formed by the occurrence of a combination of primitive and/or composite events.

Composite events increase the complexity of a base of active rules because composite events are represented by complex structures, which need to be evaluated when a composite event is raised. In the same way that a composite event increases the complexity of a base of active rules, relationships between ECA rules increase the complexity of a base of active rules.

Furthermore, active rules must be validated before its implementation into a real active database system, in order to know its behavior and to verify the presence of situations that may produce an inconsistent state in the database system.

This verification can be performed through the simulation of the active rules. In this paper an ECA rule simulator is presented, which uses a Petri net model, named Conditional Colored Petri Net (CCPN), to depict ECA rules as a Petri net structure, and with the token game animation the event occurrence and rule triggering are analyzed in order to detect active database problems such as No termination and confluence (Paton and Diaz 1999).

The remainder of the paper is organized as follows. Section 2 discusses the related work about active database systems; section 3 gives a general description of the PN model used to depict ECA rules. Section 4 describes a software development named ECAPNSim, which is used to model, simulate and analyze ECA rule sets. Section 5 shows the applicability of ECAPNSim by developing an example of ECA rule set in a bank enterprise. Finally, Section 6 concludes and gives directions for future work.

2. RELATED WORK

There are several research studies about active databases and the development of ECA rules. Relational systems, such as starburst (Widom 1996), Postgres (Stonebraker and Kemmintz 1991), Ariel (Hanson 1996), SYBASE (McGoveran and Date 1992), INFORMIX (Lacy-Thompson 1990), ORACLE (Hursh 1991), among others, provide an active functionality based on triggers, but they cannot handle composite events at all.

Triggers only supports the composite event disjunction, and structure primitive events that are defined over a table, moreover, in the action part of triggers cannot be executed another trigger.

On the other hand, Object Oriented DB systems (such as HiPAC (Dayal, Blaustein, Buchmann, Chakravarthy, Hsu, Ledin, Mc-Carthy, Rosenthal, Sarin, Cary, Livny, and Jauhari 1998), EXACT (Paton and Diaz 1999), NAOS (Collet and Coupaye 1996), Chimera (Ceri, Fraternali, Paraboschi, and Tanca 1996), Ode (Gehani and Jagadish 1996), Samos (Gatziu and Ditrich 1999)) provide more elements of active systems, like the composite event handling. Nevertheless, because of the different structures and classes used to develop Object Oriented DB systems, there is not a standard model to define ECA rules in these systems.

Few researches have adopted Petri nets as ECA rule specification language (Gatziu and Ditrich 1999), (Li, Medina-Marín, and Chapa 2002) (Schlesinger and Lörincze 1997).

Colored Petri Nets (CPN) are a high-level Petri nets which integrate the strength of Petri nets with the strength of programming languages. Petri nets provide the primitives for the description of the synchronization of concurrent processes, while programming languages provide the primitives for the definition of data types and the manipulation of their data values (Jensen 1994). So it is more suitable for active database than ordinary Petri nets since it can manipulate data values. By using CPN one can not only revealing the interrelation between ECA rules but also capture the operational semantics. For these reasons, CPN is very suitable for modeling and simulation of active rules. (Schlesinger and Lörincze 1997) adopted CPN as rule specification language, and they proposed an Action Rule Flow Petri Net (ARFPN) model, and a workflow management system was illustrated to verify their ARFPN model. However, there exists much redundant PN structure for using ”begin of”, ”end of” events, conditions and actions repeatedly. So, Their CPN model is very large even for a small rule set. Therefore, the complexity of CPN management increases. In SAMOS (Gatziu and Ditrich 1999) a SAMOS Petri Nets (S-PN) was proposed for modeling and detection of composite events. S-PN is also CPN-like where a different perspective for colors was taken. Colors in SPN are token types, and one token type is needed for each kind of primitive event; however, the framework is not Petri-net-based.

3. CONDITIONAL COLORED PETRI NET DEFINITIONS

There are several proposals to support reactive behaviors and mechanisms inside a DBS, which is best known as an ADBS. Nevertheless, these proposals are designed for particular systems, and they cannot be migrated to any other system, moreover, there is not a formal ADBS proposal.

In this paper, a general model to develop ECA rules in an ADBS is proposed, based in PN theory, which can be used as an independent engine in any DBS. An ADBS must offer both a knowledge model and an execution model. Knowledge model specifies the elements of the ECA rule, i.e., the event, condition, and action part. On the other hand, execution model describes the way in that the ECA rule set will be executed.

In knowledge model, each ECA rule element is converted into a CCPN element. The event, which activates the ECA rule, is converted in a CCPN structure that is able to perform the event detection. A Primitive event is depicted by a CCPN place, but if the event rule is composite, then the corresponding CCPN structure is generated. Both types of events finish in a place, which will be used as an input place for a transition.

A CCPN transition holds the next element of an ECA rule, the conditional part. It verifies if there are tokens in its input place and evaluates the conditional part of the ECA rule that is holding. Unlike traditional PN transitions, CCPN transitions have the ability to evaluate boolean expressions.

Finally, the ECA rule element action. When action part is executed in a DBS, it modifies the DB state. This can be viewed as an event that modifies the DB state. Events are represented as CCPN places, thus action part is represented by a place too. The difference between places for events and places for actions is that places for events are input places to transitions, and places for actions are output places from transitions.

CCPN execution model is based in the transition firing rule of PN theory. It provides mechanisms to create tokens with information, or color, about events that are occurring inside the DB. New tokens are placed in the corresponding places for those events. This is the way in that an ECA rule set is processed and both composite and primitive events are detected. By using Colored Petri Nets (CPN) is possible to depict ECA rules, but only those that have primitive events. ECA rules with composite events cannot be represented efficiently with CPN.

Conditional Colored Petri Net (CCPN) (Medina-Marín 2005b) is a Petri net extension, which inherits attributes, and transition firing rule from classical PN (Li, Medina-Marín, and Chapa 2002) (Li and Medina-Marín 2004) (Medina-Marín and Li 2005a). Furthermore, CCPN takes concepts from the CPN, such as data type definition, color (values) assignation to tokens, and data type assignation to places.

In the CPN case, data type assignation is performed for all the places of CPN, on the other hand, in the CCPN case, data type assignation for places is not general, because the CCPN handles a kind of place (virtual place – denoted as dashed circles) with the ability to hold different types of tokens. Tokens can denote different data structures, according to the tables of databases.

In order to evaluate conditional part of ECA rule stored inside a CCPN transition, a function is defined to do this task. Evaluation function analyzes the boolean expression and matches it with the DB state to determine its boolean value.

Some composite events needs to verify a time interval, hence CCPN provides a function that assigns time intervals to a CCPN transition, which will be the responsible to verify if events are occurring inside time interval defined, likewise the evaluation of ECA rule condition is done. These types of transition are named composite transition.

Each event occurs in a point of time, thus, CCPN provides a functions that assigns a time stamp to every token created. Time stamp value is the time instant in which the event has occurred. It is useful to verify if an event occurred inside a time interval or to detect composite events such as sequence and simultaneous.

Finally, every time that an event occurs, a token must be created. CCPN has a function to initialize tokens, in other words, when an event occurs in DB, a new token is created by CCPN and its attributes are initialized to the corresponding event values. The new token is put in the place that represents to detected event.

CCPN is an extension of PN that uses CPN concepts (Jensen 1994). In order to save event information in tokens and to create new tokens with data about the action part of the ECA rule, CCPN uses the concept of "color" taken from CPN. The values stored in tokens are used to evaluate the conditional part of the rule stored in the transition of CCPN. CCPN uses the multi-set concept from CPN, because a CCPN place may have several events at the same time. Unlike CPN, CCPN evaluates conditions inside transitions; meanwhile CPN evaluates conditions in its arcs.

Formally, a CCPN is defined as follows:

Definition 1. A conditional colored Petri net (CCPN) is a 11-tuple

CCPN = {Σ, P, T, A, N, C, Con, Action, D, τ, I}

where

(i) Σ is a finite set of non-empty types, called color sets.

(ii) P is a finite set of places. P is divided into subsets, i.e., $P = P_{prim} \cup P_{comp} \cup P_{virtual} \cup P_{copy}$, where P_{prim} represents primitive events and it is depicted graphically as a single circle. P_{comp} represents composite events negation, sequence, closure, last, history, and simultaneous and it is depicted graphically as a double circle. $P_{virtual}$ represents composite events conjunction, disjunction, and any and it is depicted graphically as a single dashed circle. And, P_{copy}, is a set which is used when two or more rules are triggered by the same event and it is depicted graphically as a double circle where the interior circle is a dashed one.

(iii) T is a finite set of transitions. $T = T_{rule} \cup T_{copy} \cup T_{comp}$., where T_{rule} represents the set of rule type transitions and it is depicted graphically as a rectangle. T_{copy} is the set of copy type transitions and it is depicted graphically as a single bar. T_{comp} is the set of composite type transitions and it is depicted graphically as a double bar.

(iv) A is a finite set of arcs.

(v) N is a node function. It is defined from A to $P \times T \cup T \times P$.

(vi) C is a color function. It is defined from P to Σ.

(vii) Con is a condition function. It evaluates either the rule condition if $t \in T_{rule}$ or it evaluates the time interval when $t \in T_{comp}$.

(viii) Action is an action function. It creates tokens according to action rules.

(ix) D is a time interval function.

(x) τ is a timestamp function.

(xi) I is an initialization function.

4. ECAPNSIM

Active rules development is an activity that needs to be performed carefully. Nowadays, there are few systems (Schlesinger and Lörincze 1997) which perform the analysis and debug the ECA rule base. Most of commercial ADBs (Hanson 1996), (Widom 1996) provide a syntax to ECA rule definition, however static analysis of ECA rules cannot be performed inside these systems and the ECA rule definition is only in a text way.

ECAPNSim (ECA Petri Nets Simulator) was developed under MAC OS X Server in Java. Taking advantage of Java portability, ECAPNSim can be executed in different operating systems. As an engine of ECA rules, ECAPNSim can be connected with any relational database systems such as Postgres, MS Access, Oracle, and Visual Fox Pro.

4.1 ECAPNSim architecture

ECAPNSim architecture consists of two building blocks: ECAPNSim Kernel and ECAPNSim tools environment (figure 1). ECAPNSim Kernel provides active functionality to passive database. it consists of CCPN Rule Manager, CCPN rule base, Composite Event Detector, and Rule Execution Component. ECAPNSim tools environment has a set of tools used by the ECA rule developers. Tools environment is composed by ECA rule editor, analyzer of no-termination problem, converter of ECA rules to CCPN, CCPN visualizer/editor and explanation components, termination analyzer and runtime tools.

ECAPNSim offers two modalities. In Simulation mode, users can simulate the behavior of the ECA rule base modeled by depositing tokens into the CCPN manually. And, in Real mode, the CCPN is executed by state modification of the connecting DBS.

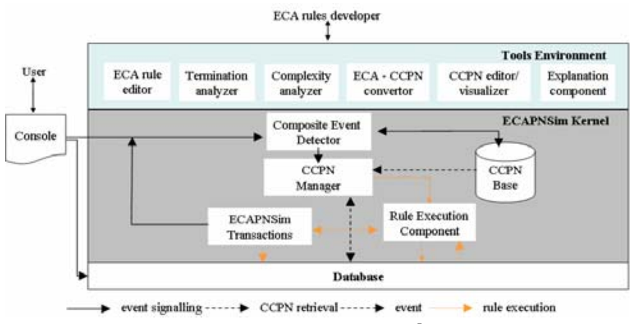


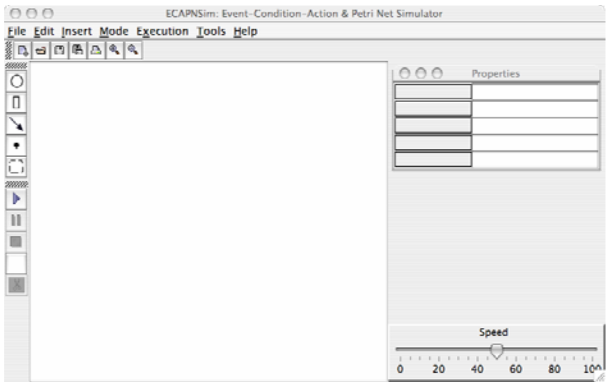
Figure 1: ECAPNSim architecture.

4.2 ECAPNSim Design

ECAPNSim offers a graphical and visual interface to represent ECA rule bases by CCPN model. Like any PN editor, ECAPNSim simulates the behavior of ECA rules by executing the CCPN model. Meanwhile simulation is running, problems like no termination and confluence can be observed obviously in the CCPN, hence ECA rule developer can modify the rule base to improve it.

The core of ECAPNSim is CCPN models. ECAPNSim contains a module to generate a CCPN structure from an ECA rule base definition written in a text file automatically. Or a CCPN model can be edited directly from a ECAPNSim user.

ECAPNSim supports CCPN design and edition from an ECA rule base, which can be moved to another position in the visualization panel. Moreover, because of there are large ECA rule bases, ECAPNSim will generate large CCPN structures, so it has zoom buttons to either increase or decrease the CCPN size. Simulation speed can be controlled through a slide. Finally, the graphical interface has tools and icons to edit a CCPN, simulate a CCPN behavior, and CCPN file management. (figure 2).


ECAPNSim: Event-Condition-Action & Petri Net Simulator
File Edit Insert Mode Execution Tools Help
Properties
Speed
0 20 40 60 80 100
Figure 2: ECAPNSim environment.

4.3 Incorporation of distribution functions

ECAPNSim was enhanced with the addition of distribution functions, which are useful to simulate and to analyze the event occurrence in an active rule base.

Distribution functions which are available in ECAPNSim are beta, binomial, Cauchy, chi square, exponential, gamma, geometric, uniform, and weibull, among others. The use of this set of functions depends on the active rule base that will be simulated.

Each place in the CCPN has the property for the definition of a distribution function, according to the frequency of the event occurrence. The values for the functions can be determined by a statistical analysis of the data about the real occurrences of the events that fire ECA rules.

Random values which are generated by distribution functions are used as inter arrival time of events in ECAPNSim. Each time an event occurrence is simulated, a token with information about that event is created, and it is putted into the place that represents the corresponding event. Hence, the token game animation is started and ECA rule developer can detect inconsistencies in ECA rule set.

5. CASE OF STUDY

In order to show the modeling of a base of active rules as a CCPN, four ECA rules are converted into a CCPN, whose description is as follows:

Rule 01 : When an employee is inserted in the office DB and the production of employee's department is modified, if the production is greater than $900.00, then the employee's bonus is updated to $100.00.

Rule 02 : When either salary or bonus of an employee is modified, if the salary is increased by more than $200.00 or the bonus is increased by more than $50.00, then the employee's rank is increased too.

Rule 03 : When the employee's rank is updated, if rank value is greater than 15, then the employee's department budget is added with $1000.00

Rule 04 : When a department budget is modified, if the budget is greater than $20,000.00, then the department production is increased of 3%.

Definition of database tables needed to this rules are as follows:

DEP(TheDep,Production,Budget).

EMP(ItsDep,TheEmp,Salary,Bonus,Rank).

CCPN obtained from the rules listed above is showed in figure 3.

From the incidence matrix showed in figure 4, and from the CCPN picture showed in figure 3, it can be observed that there exists a cyclic path in the rule firing, however, the fact that there is a cyclic in the connections of CCPN elements is not a sufficient condition to ensure that there is a no termination problem. In order to imitate the behavior of ECA rule firing according to a real situation, event occurrences are modeled through the following functions:

E0 : Insert an employee. Uniform(4,4).

E1 : Update the production value of an department. Constant value of one day.

E3 : Update employee's salary. Uniform(315,50)

E4 : Update employee's bonus. Uniform(95,17)

E6 : Update employee's rank. Uniform(103,24)

E7 : Update department budget. Uniform(365,74)

The parameters are considered in days.

The evaluation of the conditional part has probability of occurrence of 50% for true and 50% for a false result.

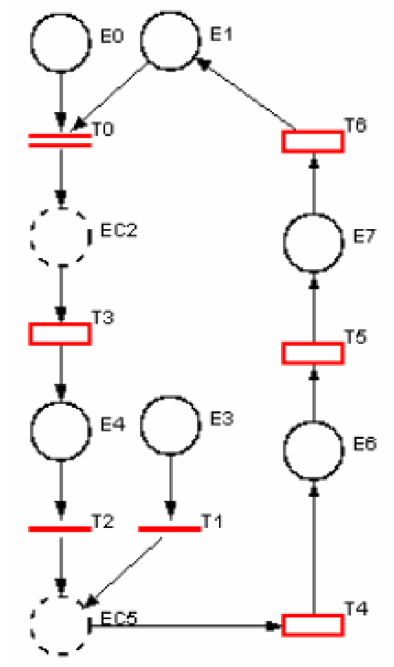


Figure 3. CCPN obtained from four ECA rules.

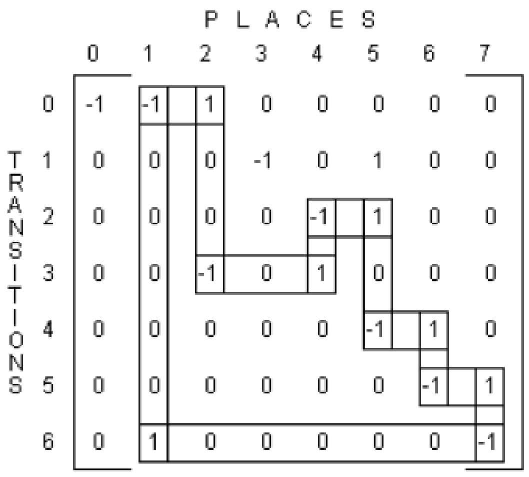
With these distribution functions assigned to each input place that represents event occurrence and running the simulation for a time corresponding to 10 years, the maximum value for the quantity of places visited by the same token was 10, i.e., the same token passed through ten places in the cyclic path, and the average value was 1.29 places visited; so this ECA rule base has the property of termination in its rule triggering.

6. CONCLUSION

Currently there are database management systems that support ECA rule definition by the use of "triggers", however "triggers" have several restrictions that limits the power that an active database must offer.

On the other side, there are research prototypes that support ECA rule definition, too; and they are more powerful because composite events such as conjunction, disjunction, etc., can de defined.

ECAPNSim is an interface that generates a CCPN from an ECA rule definition typed in the on-if-then form. It carries out the simulation of the CCPN behavior according to the event occurrence in a random way, which depends on the distribution function assigned.



		PLACES							
		0	1	2	3	4	5	6	7
TRANSITIONS	0	-1	-1	1	0	0	0	0	0
	1	0	0	0	-1	0	1	0	0
	2	0	0	0	0	-1	1	0	0
	3	0	0	-1	0	1	0	0	0
	4	0	0	0	0	0	-1	1	0
	5	0	0	0	0	0	0	-1	1
	6	0	1	0	0	0	0	0	-1

Figure 4. Incidence matrix of the CCPN showed in figure 3.

ECAPNSim has been improved with the addition of distribution functions in each place that denote an event occurrence.

A study area for active database is computer nets, where active behavior can be implemented in order to monitor traffic of LAN networks; an automatic reaction can be set by an active database system, instead of a human being monitoring. When suspicious events occur in the net, then ECA rules can be triggered and perform the corresponding action to maintain the stability in the net.

Another interesting area of application is in distributed database systems, where the development of ECA rules needs to consider the event occurrence in different servers.

There are several interesting areas where active databases are applied, mainly where systems need an automatic reaction and the response time must be immediate.

Nevertheless, ECA rule development implies that the ECA rule developer must be careful to avoid inconsistency states in the database system.

REFERENCES

Ceri, S., Fraternali, P., Paraboschi, S., Tanca, L., 1996. Active Rule Management in Chimera. In: Widom, J., Ceri, S., eds. *Active Database Systems: Triggers and Rules for Advanced Database Processing*. San Francisco CA :Morgan Kaufmann Publishers Inc., 151-176.

Collet, C., Coupaye, T., 1996. Composite Events in NAOS. *Proceedings of the 7th International Conference and Workshop on Database and Expert Systems Applications*. (*DEXA'96*). LNCS 1134, 244 - 253, September 9-13, Zurich, Switzerland.

Dayal, U., Blaustein, B., Buchmann, A., Chakravarthy, U., Hsu, M., Ledin, R., Mc-Carthy, D., Rosenthal, A., Sarin, S., Cary, M.J., Livny, M., Jauhari, R., 1988. The HiPAC Project: combining active database and timing constraints. *ACM SIGMOD RECORD*, 17 (1), 51- 70.

Gatziu, E., Ditrich, K.R., 1999. SAMOS. In: Paton, N. W., Ed. *Active Rules in Database Systems*. New York:Springer, 233-248.

Gehani, N., Jagadish, H.V., 1996. Active Database Facilities in Ode. In: Widom, J., Ceri, S., eds. *Active Database Systems: Triggers and Rules for Advanced Database Processing*. San Francisco CA :Morgan Kaufmann Publishers Inc., 207-232.

Hanson, E.N., 1996. The Design and Implementation of the Ariel Active Database Rule System. *IEEE Transactions on Knowledge and Data Engineering*. 8(1), 157-172.

Hursh, C.J., Hursch, J.L., 1991. *Oracle SQL Developer's Guide*. New York : McGraw-Hill.

Jensen, K., 1994. An Introduction to the Theoretical Aspects of Colored Petri Nets. *Lecture Notes in Computer Science: A Decade of Concurrency*, 803: 230- 272.

Lacy-Thompson, T., 1990. *INFORMIX-SQL, A tutorial and reference*. New Jersey : Prentice Hall.

Li, X., Medina-Marín, J., and Chapa, S.V., 2002. A Structural Model of ECA Rules in Active Database. *Lecture Notes in Artificial Intelligence, 2313* : 486-493.

Li, X., Medina-Marín, J., 2004. Composite Event Specification in Active Database Systems: A Petri Net Approach. *Proceedings of the IEEE International Conference on System, Man, and Cybernetics*, pp. 4885-4890. Oct 10-13, The Hague, The Netherlands.

McGoveran, D., Date, C.J., 1992. *A guide to SYBASE and SQL Server : a user's guide to the SYBASE product*, Boston : Addison-Wesley.

Medina-Marín, J., Li, X., 2005a. An Active rule base Simulator based on Petri Nets. *Proceedings of the the Third International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems MSVVEIS-2005*, pp. 96-101. May 24-28, Miami, USA.

Medina-Marín, J., 2005b. *Desarrollo de reglas ECA, un enfoque de red de Petri*, Thesis (PhD). CINVESTAV-IPN, México.

Paton, N.W., Diaz, O., 1999. Active Database Systems. *ACM Computing Surveys*, 31 (1), 64- 103.

Schlesinger, M., Lörincze, G., 1997. Rule modelling and simulation in ALFRED, *Proceedings of the 3rd. International workshop on Rules in Database Systems (RIDS'97)* (or LNCS 1312), pp. 83-99. June 26-28, Skövde, Sweden.

Silberschatz, A., , Korth, H.F., Sudarshan, S., 1999. *Database System Concepts*. *Third Ed.* New York*:* McGraw-Hill.

Stonebraker, M., Kemmintz, G., 1991. The POSTGRES Next-Generation Database Management System. *Communications of the ACM*, 34(10), 78-92.

Widom, J., 1996. The Starburst Active Database Rule System. *IEEE Transactions on Knowledge and Data Engineering*, 8(4), 583-595.

AUTHORS BIOGRAPHY

Joselito Medina-Marín. He received the M.S. and Ph.D. degrees in electrical engineering from the Research and Advanced Studies Center of the National Polytechnic Institute at Mexico, in 2002 and 2005, respectively. He is presently a Professor of the Advanced Research in Industrial Engineering Center at the Autonomous University of Hidalgo State at Pachuca, Hidalgo, México. His current research interests include Petri net theory and its applications, active databases, simulation, and programming languages.

Xiaoou Li. Received the B.S. and Ph.D. degrees in applied mathematics and electrical engineering from Northeastern University, Shenyang, China, in 1991 and 1995, respectively. From 1995 to 1997, she was a Lecturer of electrical engineering with the Department of Automatic Control, Northeastern University. From 1998 to 1999, she was an Associate Professor of computer science with Centro de Instrumentos, Universidad Nacional Autónoma de México, México City, México. Since 2000, she has been a Professor of computer science at Departamento de Computación, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV-IPN), México, City. Her research interests incluye Petri net theory and application, neural networks, information systems, data mining and system modelling and simulation.

José Ramón Corona-Armenta. He received the B.S. in Civil Engineering from Instituto Tecnológico de Pachuca in 1993, the M.S. in Engineering from the Universidad Nacional Autónoma de México in 1996, and the Ph.D. degree in Industrial Systems from Institut National Polytechnique de Lorraine in 2005. Since 2005 he has been a Professor of the Advanced Research in Industrial Engineering Center at the Autonomous

University of Hidalgo State at Pachuca, Hidalgo, México.

Marco Antonio Montufar-Benítez. He received the B.S. in Geophysical Engineering from Universidad Nacional Autónoma de México in 1985, and the M.S. in Operations Research from the Universidad Nacional Autónoma de México in 1990. Since 2000 he has been a Professor of the Advanced Research in Industrial Engineering Center at the Autonomous University of Hidalgo State at Pachuca, Hidalgo, México.

Oscar Montaño-Arango. He received the M.S. in Planning from Universidad Nacional Autónoma de México in 2000, and the Ph.D. degree Planning Systems from the Universidad Nacional Autónoma de México in 2007. Since 2006 he has been a Professor of the Advanced Research in Industrial Engineering Center at the Autonomous University of Hidalgo State at Pachuca, Hidalgo, México.

Aurora Pérez-Rojas. She received the B.S. in Industrial Engineering from Polytechnic Superior Institute José Antonio Echeverría (ISPJAE) in 1971, the M.S. in Automatic Systems from ISPJAE in 1978, and the Ph.D. degree in Technique Science from ISPJAE in 1987. Since 2006 she has been a Professor of the Advanced Research in Industrial Engineering Center at the Autonomous University of Hidalgo State at Pachuca, Hidalgo, México.