A New Approach for Event Triggering Probability Estimation in Active Database Systems to Rule Scheduling Improvement

Abbas Rasoolzadegan, Rohollah Alesheykh, Ahmad Abdollahzadeh
Intelligent Systems Laboratory
http://ce.aut.ac.ir/islab
Department of Computer Engineering and Information Technology
Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran
{Rasoolzade, Alesheykh, Ahmad}@aut.ac.ir

Abstract

Active database systems (ADBS) can react to the occurrence of some predefined events automatically. Reactive behavior of ADBS is organized by a collection of active rules. One of the most important modules of ADBS is the rule manager. The main responsibility of the rule manager is triggering, buffering, firing and selecting (scheduling) rules. Rule scheduling approach has considerable impact on performance and efficiency of ADBS. In this paper, we propose a new approach for improving the rule scheduling in ADBS. We first introduce a framework to compare and evaluate existing rule scheduling approaches. In this framework, five evaluation criteria have been proposed: Average Response Time, Response Time Variance, Throughput, Time Overhead per Transaction and CPU Utilization. Existing approaches have been evaluated by using this framework and the approach which has the most positive impact on performance and efficiency of ADBS has been selected by analyzing the weaknesses and strengths of existing approaches. Then, to improve the selected rule scheduling approach, we developed an Event Triggering Probability Estimation algorithm and integrated this algorithm to selected rule scheduling approach. Results of experiments show that the new proposed algorithm increases the positive impact of selected rule scheduling approach on performance of ADBS.

1. Introduction

Traditional database systems (passive databases) can only store and retrieve data [1]. They work directly with user requests. By application getting grown and databases getting larger, passive database systems were unable to manage these large systems, and as a result, database systems needed to supervise some predefined special situations and react to some occurrences [2]. These predefined situations are called events. For supporting reactive behavior of new database systems (called Active Database Systems), Event-Condition-Action (ECA) rule format was created. The ECA format has three sections: Event, Condition, and Action. When an event occurs, condition gets evaluated and if the condition is true, action is executed.

1.1. Active Database Management Systems (ADBMS)

There are many types of events in different ADBS, such as data insertion, data manipulation, transaction start, commit, abort and rollback. The conditions specify what should be checked after the occurrence of an event and before execution of an action. The actions can contain anything such as: data modification and retrieval (in relational DBMS), transaction operation like commit or abort, method invocation (in object-oriented DBMS), procedure call (in relational DBMS) and rule operations [3].

One of the most important aspects of the ADBS that affects their power is rule language [4]. Besides the main components of rule language, there are additional features that play an important role in the specification of ECA rules. One of these important features is coupling modes. The phases of rule execution discussed so far are not necessarily executed contiguously, but depend on the so-called coupling modes which are pairs of values (x, y) associated with each rule. The value ‘x’ couples event signaling and condition evaluation of a rule, whereas ‘y’ couples condition evaluation and action execution. Possible coupling modes are immediate, deferred and independent [4]:

- Immediate mode: in this mode, when an event occurs, current transaction is suspended and the action is executed, if the condition holds.
- Deferred mode: in this mode, after the occurrence of an event condition evaluation and action execution is deferred till the end of the current transaction. In deferred mode, the action of triggered rule should be executed before current transaction commits.
- Independent mode: when an event is triggered in independent mode, there are no time-constraints and restrictions on condition evaluation and action execution.

In order to support reactive behavior of the ADBS, they should contain additional units for managing rule base and rule processing steps in comparing with Passive Database Management systems. Such a DBMS is called an Active Database Management System (ADBMS) [5].

1.2. Rule Processing In Active Database Systems

In this section, we briefly describe what happens when an event occurs. An application runs sequentially until an event occurs. After an event occurs, the rule processing unit is activated and triggers the appropriate rule(s). Triggered rule(s) are queued in a temporary buffer. Then triggered rule(s) are selected according to some special criteria and then their "condition" section is evaluated. If condition is true, the action section will be executed. If the current rule triggers some other rules, new triggered rules will be passed to the rule processing unit. When there aren't any triggered rules, the application continues running. In summary, there are five different rule processing steps:

(a) Event Signaling: When a primitive event occurs, the primitive event detector signals that event. Additionally, the composite event detector considers these primitive events that contribute to composite events.

(b) Rule Triggering: After the event is signaled, ECA rules that correspond to the signaled event are selected, and for each of them rule instances are created. In each rule instance, there is some additional information based on scheduling approach, such as timestamp, deadline, execution time, etc. These rule instances are buffered to use in the next step.

(c) Condition Evaluation: After the buffering of rule instances, their conditions are evaluated. Then, for each rule with a true condition, a transaction is generated.

(d) Transaction Selection: This step is also called transaction scheduling phase. In this phase, a selection algorithm [6] operates on execution buffer and selects one transaction which is generated based on triggered rules, and sends the transaction to the execution unit.

(e) Transaction Execution: Transactions generated based on triggered rules are executed in this phase.

This paper has five sections. In section two, we analyze existing rule scheduling approaches in ADBS. In section three, we introduce a framework to compare and evaluate existing rule scheduling approaches. In this framework, five evaluation criteria have been proposed: Average Response Time, Response Time Variance, Throughput, Time Overhead per Transaction and CPU Utilization. Existing approaches have been evaluated by using this framework and the approach which has the most positive impact on performance and efficiency of ADBS has been selected by analyzing the weaknesses and strengths of existing approaches. Then in section four we introduce an Event Triggering Probability Estimation algorithm and integrate it to selected scheduling approach. Then we show the positive impact of this algorithm on performance of ADBS. At the end, in section five, there is a conclusion and some favorite jobs we tend to do in the future for improvement more rule scheduling in ADBS.

2. Related Works

In this section we briefly describe the approaches used for rules scheduling. For selecting one of the buffered rules the execution unit uses a selection algorithm. There are numerous approaches for rule scheduling in ADBS [7]. Rule processing in ADBS has an active nature [7]. This means that each triggered rule might cause other rules to fire. So ADBS can not use typical scheduling approaches.

In the rest of this section we briefly describe rule processing approaches used in ADBS. In this paper, we use "rule selection" and "transaction selection" terms interchangeably.

- **Random Scheduling Approach:** Random selection is one of the easiest approaches for rule scheduling in ADBS [10]. This approach has been implemented in RPL and Ode active database systems [10]. In the Random approach ADBMS selects one of the activated rules randomly. The most important characteristic of this approach is its simplicity, at the cost of efficiency.

- **Static Priority Scheduling Approach:** In this approach, the system assigns a numeric priority to each ECA rule but the priorities need not be unique. In the Ariel [11] and POSTGRES [3] systems, each rule is assigned a priority between -1000 and +1000. When an activated rule should be selected to run, the rule that has the minimum static priority is selected.

- **FCFS Scheduling Approach:** FCFS (First Come First Serve) scheduling approach is one of the classic approaches used for rule scheduling in ADBS [6]. When an event occurs and rules are triggered, an instance of each triggered rule is generated. This instance of triggered rule contains a timestamp which shows the time the rule is triggered. When an activated rule should be selected to run, the activated rule that has the earliest timestamp is selected. This scheduling approach is used in SAMOS [12].

- **Concurrent Execution Scheduling Approach:** HiPAC active database system [6] supports this approach. Rule processing in HiPAC is invoked whenever an event occurs and triggers one or more rules. This approach differs from most other rule scheduling approaches in its handling of multiple triggered rules. So we can not quantatively compare it with other rule scheduling approaches which are serial. HiPAC executes all triggered rules concurrently. This means that if during rule execution, additional rules are triggered, they are executed concurrently. Another ADBMS called FAR [8] also uses concurrent rule execution.

- **EDF based Scheduling Approach:** Earliest Deadline First (EDF) is one of the classic algorithms for transaction scheduling in real-time systems [7]. The EDF based approach [7] is one of the best approaches introduced for rule scheduling till now. This approach has been presented for Real-time Active Database (RADB). In this approach rules are scheduled based on their deadline. This approach has three different versions: (1) EDF_{PD}, (2) EDF_{DIV}, and (3) EDF_{SL}. The EDF_{PD} is a static baseline policy where rules priorities do not change with time. EDF_{DIV} and EDF_{SL} are dynamic policies where rules priorities change depending on the amount of dynamic work they have generated [7].

- **E_x-SJF Scheduling Approach:** The SJF algorithm is one of the classic scheduling algorithms. This approach is one of the most effective scheduling approaches [13]. SJF algorithm is not useful for rule scheduling in ADBS due to active work load nature [7] of it. So there is defined preprocess for preparing rule base to use the SJF algorithm for rule scheduling in E_x-SJF (Extended SJF) approach [9]. The difference between SJF and E_x-SJF is in manner of transactions (rules) execution time calculation. In E_x-SJF approach, the execution time of each parent transaction (rule) is related to the number of its immediate and deferred child transactions (rules). According to manner of interference of immediate and deferred child transactions (rules) execution time in their parent rules execution time, there are two versions of E_x-SJF which are named E_x-SJF_{EXA} and E_x-SJF_{PRO} [9].

Although E_x-SJF is generally more effective than other mentioned rule scheduling approaches, it has some weakness points such as: (1) It is useless in real-time systems and systems with concurrent execution ability. (2) It does not calculate the execution time of transactions (rules), exactly.

3. Proposed Framework to compare and evaluation of existing rule scheduling approaches

In this section we introduce a framework for comparison and evaluation of existing rule scheduling approaches [14]. This framework contains five evaluation criteria: Average Response Time, Response Time Variance, Throughput, Time Overhead per Transaction and CPU Utilization.

We need an environment which can simulate an active database system. With such system we can implement each rule scheduling approach and consider the performance of it. For this reason, an environment which is named Active Database System Simulator (ADSS) has been designed and implemented at the Intelligent Systems Laboratory [9].

Experiments are performed in three modes [9]: (1) "Deferred mode", (2) "Immediate mode" and (3) "Composite mode". In the first mode system uses rules only in deferred mode. In the second mode, system uses rules only in immediate mode and ultimately in the third mode, system uses rules in all immediate, deferred, and independent modes. Results of experiments in deferred, immediate and composite modes are shown in tables 2, 3, 4, respectively. The content of each cell shows the rank of corresponding scheduling approach according to corresponding evaluation criteria.

Table 2. Results of simulation of available rule scheduling approaches in deferred mode

Approaches \ Evaluation Criteria	Average Response Time	Response Time Variance	Throughput	Time Overhead per Transaction	CPU Utilization
Random	4	3	1	1	1
Static Priority	3	2	3	2	1
FCFS	3	2	3	2	1
EDF_{PD}	2	1	4	2	1
EDF_{DIV}	2	1	4	2	1
EDF_{SL}	2	4	5	2	1
E_x-SJF_{EXA}	1	1	2	2	1
E_x-SJF_{PRO}	1	1	2	2	1

Table 3. Results of simulation of available rule scheduling approaches in immediate mode

Approaches \ Evaluation Criteria	Average Response Time	Response Time Variance	Throughput	Time Overhead per Transaction	CPU Utilization
Random	2	3	1	1	3
Static Priority	2	3	1	1	3
FCFS	2	2	1	1	3
EDF_{PD}	2	5	2	1	3
EDF_{DIV}	2	2	2	1	3
EDF_{SL}	2	4	3	1	1
E_x-SJF_{EXA}	1	4	2	1	2
E_x-SJF_{PRO}	1	1	2	1	2

Table 4. Results of simulation of available rule scheduling approaches in composite mode

Approaches \ Evaluation Criteria	Average Response Time	Response Time Variance	Throughput	Time Overhead per Transaction	CPU Utilization
Random	3	3	2	3	3
Static Priority	3	3	2	3	3
FCFS	3	3	2	1	1
EDF_{PD}	2	2	1	1	2
EDF_{DIV}	2	2	1	1	2
EDF_{SL}	1	1	3	2	2
E_x-SJF_{EXA}	2	2	3	3	2
E_x-SJF_{PRO}	1	1	4	2	2

Results of experiments show that E_x-SJF_{PRO} has generally the most positive impact on performance (Response Time, Response Time Variance, and Throughput) and efficiency (Time Overhead per Transaction and CPU Utilization) of ADBS, but it has also some weaknesses as stated previously. In the next section we introduce a solution to solve one of its problems.

4. A new approach for Event Triggering Probability Estimation

In section 3 we concluded that E_x-SJF_{PRO} is the most effective approach based on mentioned evaluation criteria in any of the three referred modes. But as mentioned in the last section, this approach has also some weakness points such as disability in exact rule execution time calculation. Before introducing the new approach for Event Triggering Probability Estimation which solves this problem, we state the manner of rules execution time calculation in E_x-SJF_{PRO} approach [9].

As we mentioned in section one, there are three coupling modes for rule triggering. Figure 1 shows the life of a complex active transaction T that triggers some transactions in immediate and deferred modes. Active transactions in ADBS are produced by active rules.

Transaction T arrives at time t_1 (a(T)) and is started at time t_2 (s(T)). Deferred transactions T_1^{def} and T_2^{def} are triggered at times t_3 and t_4, respectively. Immediate transactions T_1^{imm} and T_2^{imm} are triggered at times t_5 and t_7, respectively. This figure shows that the transactions T_1^{imm} and T_2^{imm} are executed immediately while T is suspended. Finally, figure 1 shows that once the T is completed at t_9, the deferred transactions T_1^{def} and T_2^{def} are executed. We refer to transaction T as parent transaction and refer to immediate, deferred and independent transactions activated by T as child transactions. Also, we refer to active rules that produce the transaction T as parent rules and refer to active rules triggered by the parent rule as child rules. Child rules can produce immediate, deferred and independent child transactions.

According to figure 1, the execution of transaction T is finished at time t_9 but it is committed at time t_{10}. In other words, the execution time of each parent transaction (rule) is related to the number of its immediate and deferred child transactions (rules). Therefore if we can determine the relations between rules before the execution we can compute the exact execution time of each rule. For the extraction of relations between rules in the rule base, a hierarchical structure of rules is defined and is referred to as "Rule Execution Tree" [9].

The E_x-SJF approach has three steps:

Step 1: Scanning the rule base.

Step 2: Constructing "Rule Execution Tree".

Step 3: Computing "exact execution time" for each rule using post order traversal.

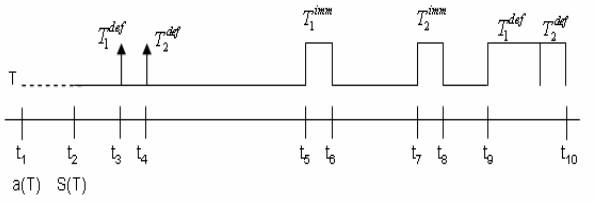

Figure 1. Life of a complex active transaction [9]

4.1. Scanning the rule base

Each active rule in the rule base is defined using the ECA format [5]. Each active rule is considered as $R_i(E_i,C_i,A_i,ES_i,RS_i)$ where E_i,C_i and A_i are event, condition and action of rule R_i, respectively. The ES_i is a set of events occurred by the execution of A_i instructions and the RS_i is a set of rules activated by one or more events of the ES_i set. For constructing the "Rule Execution Tree" (RET) two ES and RS sets are added to each active rule.

4.2. Constructing the RET and calculating exact execution time

Each rule in ADBS is executed if and only if it is activated and its condition is true. Therefore, we consider a new parameter to compute the "exact execution time" of active rule. This new parameter is the probability of the rule's condition being true. The condition part of active rules may be formed as a combination of logical conditions using logical operations such as AND, OR, NOT, etc. For computing the probability of the condition part of each rule which is formed as $C_1 \cup C_2 \cap C_3 \ldots \cup C_n$, we assume that:

$C_1, C_2, C_3, \ldots C_n$ are independent.

$P(C_1) = P(C_2) = P(C_3) = \ldots = P(C_n) = 1/2$.

Table 5 shows the parameters which are used for calculating "exact execution time" of rules. Figure 2 shows a generic RET. Each rule in figure 2 is presented with three attributes: R_i (the name of rule), X_i (rule execution time), P_i (the probability of condition part of the rule). Now the "exact rule execution time" of each rule is calculated as shown in equation 1 [9], where 'n' is the total number of levels of RET and 'I' is the level of the rule 'R' in the RET.

For calculating the exact execution time of each rule, equation 1 should be evaluated from the bottom of tree to the top (from leaf to root). Therefore, post order traversal algorithm is used to make equation 1 applicable. Then the "exact execution time" of each active rule is calculated and stored in X attribute presented in figure 2.

Table 5. The parameters for computing "the exact execution time" of active rules

$P(R)$	The probability of condition part of rule R to be true
$X^{imm}(R_i)$	The exact execution time of rule Ri which is triggered in immediate mode
$X^{def}(R_j)$	The exact execution time of rule Rj which is triggered in deferred mode
$L(R)$	The execution time of rule R
$n^{def}(R)$	Number of deferred rules triggered by T during its execution
$n^{imm}(R)$	Number of immediate rules triggered by T during its execution
$X(R)$	The exact execution time of rule R

Now if we use these active rules in an ADBS, SJF algorithm could be used for rule scheduling. This

approach is named Extended SJF Probabilistic (E_x-SJF_{PRO}).

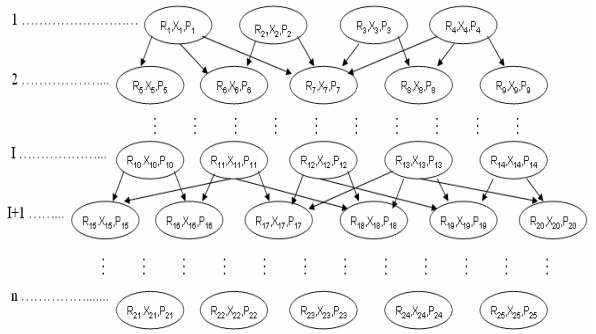

Figure 2. A Generic Rule Execution Tree [9]

$$X(R^I) = L(R^I) + \sum_{i=1}^{n^{imm}(R^I)} P(R_i^{I+1}) * X^{imm}(R_i^{I+1}) + \sum_{j=1}^{n^{def}(R^I)} P(R_j^{I+1}) * X^{def}(R_j^{I+1})$$
$$I = \{n-1, n-2, ..., 1\} \quad (1)$$

As mentioned before, the correctness probability of each logical statement in condition part of each rule is assumed ½ that is not exact. Calculation of logical statement correctness probability more exactly leads to more exact rule execution time calculation, and increases the efficiency of the scheduling approach.

Proposed solution adds an estimation module to E_x-SJF_{PRO} which is introduced as a new version of it and we name it E_x-SJF_{PRO}-V.1.8. In new approach, at first, execution time of each rule is calculated in the E_x-SJF_{PRO} approach. Then in every time which an activated rule such as R_1 is selected according to scheduling approach from activated rules list for condition evaluation, correctness probability of each logical statement of condition part of R_1, i.e. $P(R,LS_i)$ is repeatedly calculated and stored based on equation 2.

$$P(R, LS_i) = \frac{T1}{T2} \quad (2)$$

(T_1: times which LS_i has been evaluated and has been true so far, T_2: times which R has been activated so far.)

This process is repeated for each logical statement of condition part of the corresponding rule until the changes rate of correctness probability of that logical statement, achieves to a desired value (e.g., 0.0000001 and in general mode ε). At this time, new value is replaced with primary default value (i.e. ½). It is evident that the smaller value ε, the more exact calculation of $P(R,LS_i)$. When correctness probabilities of all logical statements of a condition are updated, correctness probability of that condition is updated too. And ultimately execution time of rule R is updated, when condition part correctness probability and all R's childes execution time are updated. So, after passing a short time from start of executing the system (which this time is very insignificant in compare with total executing time of system), execution time of all rules are calculated with a satisfied precision (which amount of this precision is depend on value ε).

An important characteristic of ADSS is flexibility. It means that we can implement each rule scheduling approach, only by replacing the selection algorithm in the ADSS. So we implemented the E_x-SJF_{PRO}-V.1.8 approach and embedded it in ADSS. Then we compared E_x-SJF_{PRO} and E_x-SJF_{PRO}-V.1.8 approaches in deferred, immediate, composite modes.

Table 6 shows the percentage of improving the rule scheduling in new version in compare with E_x-SJF_{PRO}, based on three evaluated criteria: average response time, response time variance and throughput

Table 6. Percentage of rule scheduling improvement in E_x-SJF_{PRO}-V.1.8 in compare with E_x-SJF_{PRO}

Evaluation criteria / Mode	Average Response Time	Response Time Variance	Throughput
Immediate	8%	12.6%	9%
Deferred	18.8%	33.6%	14.3%
Composite	14.2%	21.4%	12%

Results of experiments show that by adding estimation module, executing time of rules are calculated more exactly (the amount of this precision depends on value ε) and leads to improving the Average Response Time, Response Time Variance and Throughput of E_x-SJF_{PRO} approach. Event triggering probability estimation process dose not impose any overhead on the ADBS. So the Time Overhead per Transaction and CPU Utilization of E_x-SJF_{PRO}-V.1.8 and E_x-SJF_{PRO} are equal.

5. Conclusions and future works

In this paper, we first introduced a framework to compare and evaluate existing rule scheduling approaches. In this framework, five evaluation criteria had been proposed: Average Response Time, Response Time Variance, Throughput, Time Overhead per Transaction and CPU Utilization. Existing approaches were evaluated by using this framework and the approach which has the most positive impact on performance and efficiency of ADBS was selected. Then, to improve the selected rule scheduling approach, we developed an Event Triggering Probability Estimation algorithm and integrated this algorithm to selected rule scheduling approach. Results of experiments show that the new proposed algorithm increases the positive impact of selected rule scheduling approach on performance of ADBS.

In future, we intend to increase the efficiency of E_x-SJF_{PRO}-V.1.8 approach by adding a learning module (using techniques such as Learning Automata and Reinforcement). Learning techniques cause to calculate the value ε automatically. System calculates the value ε according to feedback which is received from functionality of the system. Functionality of the system is determined based on system's status according to evaluated criteria.

Acknowledgment

The authors would like to thank Mr. Mohammad Reza A. Shirazi for his technical assistance and helpful advices.

References

[1] C. J. Date, “An Introduction to Database Systems”, 8th Edition, Addison-Wesley, 2003.

[2] E. Bertino, B. Catania and G. P. Zarri, "Intelligent Database Systems", Addison-Wesley, 2001.

[3] S. Potaminsto and M. Stonebraker, "The POSTGRES Rule System", in Active Database Systems: Triggers and Rules for Advanced Systems, Morgann Kaufmann Publishers, Sanfrancisco, CA, 1996.

[4] H. Fritshi and Z. Flaach, “A Component Framework to Construct Active Database Management Systems”, PhD Thesis, CS Department, University of Zurich, 2002.

[5] Vadua, "Rule Development for active database", PhD Thesis, CS Department, University of Zurich, 1999.

[6] Vaduva, S. Gatziu and K. R. Dittrich, "Investigating Termination in Active Database Systems with Expressive Rule Languages". In Proceedings of the 3rd International Workshop on Rules In Database Systems, pp. 149-164, Skovde(Sweden), 1997.

[7] R. M. Sivasankaran, J. A. Stankovic, D. Towsley, B. Purimetla and K. Ramamritham, "Priority Assignment in Real-Time Active Databases", The International Journal on Very Large Data Bases, Vol. 5, No. 1, January 1996.

[8] S. Ceri, C. Gennaro, S. Paraboschi and G. Serazzi, “Effective Scheduling of Detached Rules in Active Database”, IEEE Transaction on Knowledge and Data Engineering, Vol. 15, No.1, January/February 2003.

[9] R. Alesheykh, A. Abdollahzadeh, "Evaluation of Shortest Job First Approach for Rule Scheduling in Active Database Systems", in the Proceedings of the 9th IASTED International Conference on Artificial Intelligence & Soft Computing (ASC'05), Benidorm, Spain, September 2005.

[10] S. Chakravarthy, "Architectures and monitoring techniques for active databases: An evaluation". In Technical Report TR-92-041, University of Florida, Gainesville, 1992.

[11] J. Stankovic, M. Spuri, M.D. Natale and G.C. Buttazo, ”Implications of Classical Scheduling Results for Real-Time Systems”, IEEE Computer Society Press, Los Alamitos, CA, 1995.

[12] Geppert, S. Gatziu, K. R. Dittrich, H. Fritschi, and A. Vaduva, "Architecture and implementation of the active object-oriented database management system SAMOS", Technical Report 95.29, CS Department, University of Zurich, 1995.

[13] H. Theodore, "A survey of Active Database Systems", available in http://www.doc.ic.ac.uk/~twh1/academic/papers/active.ps, April 1997.

[14] R. Alesheykh, A. Abdollahzadeh, "Evaluation and Comparison of Rule Scheduling Approaches in Active Database Systems", in the Proceedings of the 2nd IASTED international Multi-Conference on Automation, Control, and Information Technology (ACIT’05), Russia, June 2005.