An Integrated Data Management Approach to Manage Health Care Data [*]

Diogo Guerra[1], Ute Gawlick[2], Pedro Bizarro[1], Dieter Gawlick[3]

CISUC/University of Coimbra[1], University of Utah Health Sciences Center[2], Oracle Corporation[3]

deag@student.dei.uc.pt, ute.gawlick@hsc.utah.edu, bizarro@dei.uc.pt, dieter.gawlick@oracle.com

Abstract: Surgical Intensive Care Unit data management systems suffer from three problems: data and meta-data are spread out in different systems, there is a high rate of false positives, and data mining predictions are not presented in a timely manner to health care staff. These problems lead to missed opportunities for data analysis, alert fatigue and reactive, instead of proactive analysis. In this demo, and in contrast to current CEP efforts, we present a proof-of-concept, integrated engine that runs entirely within a single database system. The resulting novel and low cost event processing architecture uses features and components commercially available from Oracle Corporation. We demonstrate how multiple data from a real-world surgical intensive care unit (bed-side sensors and all other information available about the patients) are assimilated and queries, alarms, and rules are applied. The system is highly customizable: staff can point and click to create, edit and delete rules, compose personal rules (per patient, per doctor, per patient-doctor), and, while maintaining a hierarchy of rules, create rules that inherit and override previous rules. The system is also integrated with the data mining module, being able to offer predictions of high risk situations in real-time (e.g., predictions of cardiac arrests). Using simulated inputs, we show the complete system working, including writing and editing rules, triggering simple alerts, prediction of cardiac arrests, and visual explanation of predictions.

1 Introduction

Modern medical institutions have electronic devices that continuously monitor the vital signs of patients such as heart rate, cardiac rhythm, blood pressure, oxygen saturation and many others. Those devices are usually set to trigger alerts for critical values that are above or below predefined thresholds. Due to the static configuration of those thresholds, the devices alert the doctors and nurses for the same critical values regardless of patient condition, demographics, and alarm history. That is, the monitoring middleware typically does not distinguish between:

- A patient with a cardiac condition and a patient without a cardiac condition;

[*] A shorter version of this paper has been published at DEBS2009. The presentation at BTW 2011 discusses the work described in this paper as well as the follow on work presented at HealthInf 2011 [6]

- A male baby with high heart rate and a female senior with high heart rate;
- A patient that started to have high fever and a patient that has had high fever more than 30 minutes.

Ignoring those differences leads to missed alarms, a very high rate of false alarms, alert fatigue and causes medical staff to ignore most of the alerts.

Another problem is that patient information is normally spread out in multiple independent systems. Those systems contain details of exams such as x-rays, MRI's, blood tests or unstructured free text entered by nurses and doctors. The dispersion of information through independent systems does not allow the systems to make decisions considering the complete patient information.

Data mining systems can detect patterns to predict or identify serious conditions but are normally used offline and in different systems. ICUs are time critical environments where data mining operations (scoring) to identify possible risk situations must be performed continuously.

1.1 Contributions

The goal of this prototype, based on the Surgical Intensive Care Unit environment of the University of Utah Health Sciences Center, was to build a system with the following characteristics:

- A single integrated persistent system to manage historical data, events, rules, and data mining models. See Section 2.
- A highly customizable system that: lets users edit and create rules, maintains a hierarchy of rules, and allows personal rules and complex composable rules, thus contributing to reduce alert fatigue. See Section 3.
- A system able to identify possible future risks by per-forming data mining in soft-real-time. See Section 4.

2 Components and Architecture

This prototype was developed with Oracle technologies. Some of the technologies used in this prototype (Total Recall, DCN, Rules Manager, Data Mining) have been developed as independent features and not as components of an integrated system for event processing. Part of the challenge and motivation to build this demo was using these technologies to build an integrated system with the characteristics identified above. These technologies and their contribution for the presented prototype are described below.

2.1 Oracle Total Recall (TR)

In a normal database, if you want to store the current and historical values of a sensor, you normally use application logic and write the values time stamped, and possibly in different tables for the actual and historical values. Thus, a sensor with 100 readings takes 100 records in the table(s).

However, using Oracle TR [3] (a technology developed to handle the versioning of records), the reading will take just one record in the table and the other 99 historical values will be transparently kept elsewhere. Those historical versions can be accessed with the AS OF and VERSIONS BETWEEN clauses. The AS OF clause is used to select values in some point in time:

```
-- Select temperature 5 mins ago
SELECT temp FROM patients
   AS OF TIMESTAMP
   systimestamp – INTERVAL ‘5’ MINUTE);
```

The VERSIONS BETWEEN clause queries past ranges:

```
-- Select temp of patient in the last hour
SELECT temp FROM patients
   VERSIONS BETWEEN TIMESTAMP
  (systimestamp – INTERVAL ‘1’ HOUR) AND
   systimestamp
WHERE pid = 10;
```

With TR a slightly extended SQL can be used to access the history; the maintenance of history records is transparent.

2.2 Continuous Query Notification (CQN)

Oracle CQN [2] is a technology that allows the database engine to notify clients about new, changed or deleted data. CQN is different from database triggers. While triggers fire when SQL statements are executed, CQN only notifies about committed data; in one case you see dirty data in the other case you see *committed* data. This technology has two major modes of operation: Object Change Notification and Query Result Change Notification. Query Result Change Notification (the option used in this prototype) allows the user to define with a query what changes should be notified. For example, the user can use CQN to watch for changes in the temperature of patient number 10:

```
SELECT temp FROM patients WHERE pid = 10;
```

CQN will notify only if the temperature of patient number 10 is different at the end of the transaction.

2.3 Oracle Rules Manager (RM)

Oracle RM [1] is a rules engine inside the database. It works based on events that are represented by objects and matches those events to rules defined by the user. The user can define rules that identify, e.g., sequence of events, patterns based on aggregations over time- or count-based windows, non-occurrence of events, disjunction of conditions, or using user-defined functions. RM can watch for changes in tables itself or, as done here, can accept events through its API (accessible from PL/SQL or externally using JDBC and similar interfaces).

RM can define highly complex scenarios without defining very complex rules by using user defined callbacks, production of events to be caught by other rules (see Section 3.1), user-defined conflict resolution policies and user-defined custom event consumption policies. All these features make the system very customizable.

The rules are stored in a table (simplifying rule maintenance) and defined either as a WHERE clause or in a XML specific language as show in Figure 4.

2.4 Oracle Data Mining (ODM)

Oracle DM [4] is data mining engine embedded in the Oracle Database. The users can run data mining algorithms and also build and run models over the data all within the database.

In this project, the data mining algorithms take as input the patient laboratory information and output the predicted risk of patients having a cardiac arrest or respiratory failure in the next 24 hours. The two models, built by Pablo Tamayo from Oracle/MIT, use a 725 patient training set from the SICU of the University of Utah Health Sciences Center.

2.5 Architecture

The goal of the prototype was to develop a single, integrated system able to perform different kinds of processing in a centralized and integrated system.

All the information persists in database tables. These tables are TR enabled and automatically keep history of changes for each record (See 1 in Figure 1). CQN monitors changes on sensor states (2 in Figure 1), generates corresponding events objects and sends them to be consumed by RM by calling its add_event function (3 in Figure 1). Then, the RM evaluates the rules and triggers actions to alert doctors and nurses through dashboards and mobile device communication channels (4 in Figure 1). Some rules also trigger calls to data mining models (5 in Figure 1) to apply predictions in real time. The results are sent back to the Rules Manager (6 in Figure 1), which then evaluates new rules to determine if those results need to be pushed to medical staff as well. Thus the data mining operations will only be applied in situations that will likely give some interesting results and not every time the data changes. As described next in Section 3, the alert system is highly customizable: medical staff can create, edit rules, and delete and even create new rules that inherit information and override other existing rules.

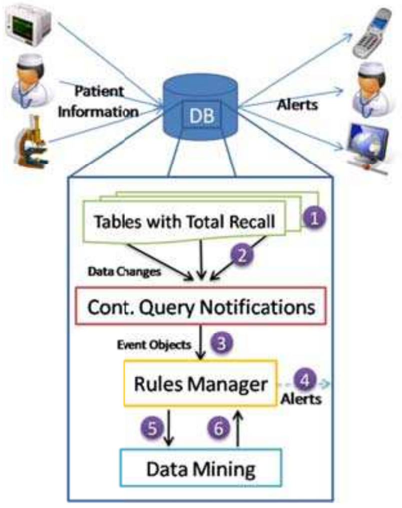
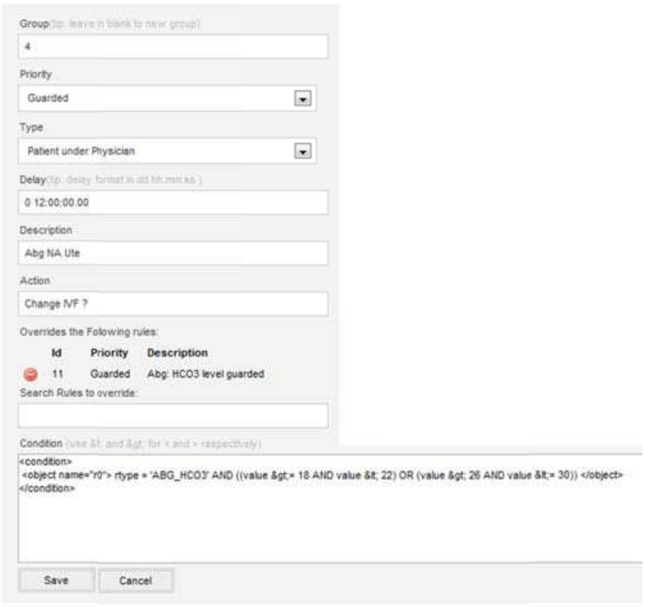


Fig. 1 - Architecture

3 Customization

The customization of rules was a must-have requirement, made possible by Rules Manager. In the alerts section a physician can edit, add or delete alerts. The alerts have multiple priorities: can be applied to all patients, only for one patient, for all patients of a specific physician or only for a specific patient and physician pair. The alerts can relate multiple values, can watch for a state during a specified time, and can also be set to not alert during some time after the first alert to reduce alert fatigue. In Figure 2, the physician overrides a general rule for Sodium level in the guar-ded priority with a new range. This alert will only trigger for this patient under the supervision of this physician.

For the basic vitals, the ones that can be most frequently needed for alerts, the physician can change their alerts easily with help of sliders to define ranges. This alert will only apply to that patient and for the physician that changed the alert values. To help determine the most appropriate values to the patient, a chart with the history of that vital will appear as shown in Figure 3.



Group (tip: leave in blank to new group)

4

Priority

Guarded

Type

Patient under Physician

Delay (tip: delay format in dd hh:mm:ss)

0 12:00:00.00

Description

Abg NA Ute

Action

Change IVF ?

Overrides the Folowing rules:

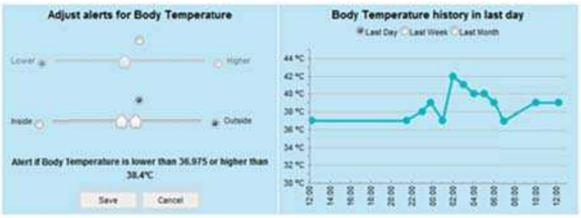
Id	Priority	Description
11	Guarded	Abg: HCO3 level guarded

Search Rules to override:

Condition (use &lt; and &gt; for < and > respectively)

```
<condition>
 <object name="r0"> rtype = 'ABG_HCO3' AND ((value &gt;= 18 AND value &lt; 22) OR (value &gt; 26 AND value &lt;= 30)) </object>
</condition>
```

Fig. 2 – Form to edit / add alerts



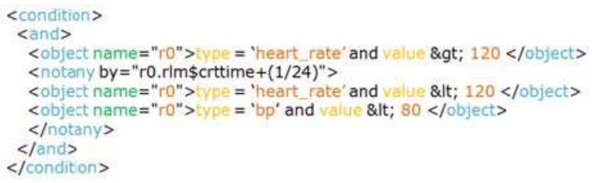
Alert if Body Temperature is lower than 36.975 or higher than 38.4°C

Fig. 3 - Definition of alerts for vitals

3.1 Composability of Rules

The system success depends mostly on the way how rules can be expressed and how complex scenarios can be implemented. Rules Manager has a great support to design complex scenarios based on simple rules and patterns.

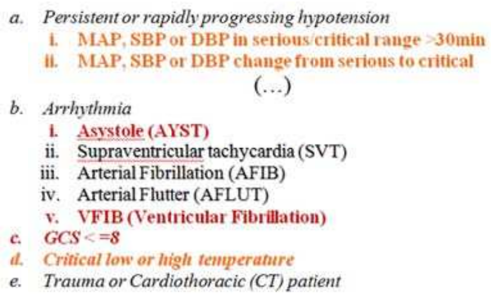
The rules are written in XML and allow relating events and performing aggregations over windows and also specifying situations of non events. Many rules could be defined with the language, but is beyond the scope for this demonstration. As example, the rule that checks for an occurrence of a heart rate higher than 120 for more than 1 hour and also that the bp is higher than 80 in the last hour is:



```
<condition>
  <and>
    <object name="r0">type = 'heart_rate' and value &gt; 120 </object>
    <notany by="r0.rlm$crttime+(1/24)">
    <object name="r0">type = 'heart_rate' and value &lt; 120 </object>
    <object name="r0">type = 'bp' and value &lt; 80 </object>
    </notany>
  </and>
</condition>
```

Fig. 4 - Rule example

By composing rules, it is possible to define more complex scenarios than the ones that the language allows in a simple rule. Figure 5 shows a piece of a scenario description in medical technical language to identify possible cardiac arrest situations. The complete rule takes more than a page to be described in technical English and can be implemented in Rules Manager with about 50 rules.



a. *Persistent or rapidly progressing hypotension*
   i. **MAP, SBP or DBP in serious/critical range >30min**
   ii. **MAP, SBP or DBP change from serious to critical**

(…)

b. *Arrhythmia*
   i. **Asystole (AYST)**
   ii. Supraventricular tachycardia (SVT)
   iii. Arterial Fibrillation (AFIB)
   iv. Arterial Flutter (AFLUT)
   v. **VFIB (Ventricular Fibrillation)**

c. ***GCS <=8***

d. ***Critical low or high temperature***

e. *Trauma or Cardiothoracic (CT) patient*

Fig. 5 – Technical definition of a complex rule

The colors represent different weights and a complex alert should occur when one red or three orange or five black statements are true. This is possible using an event approach where each statement is implemented as a simple rule that when is evaluated as true inserts a signaling event into the system with its priority (color). Other rule will be monitoring for those types of events and applies the general rule.

4 Integration with Data Mining

Another feature of the system is the integration of data mining models that score the status of the patients for the probability of risk situations like Cardiac Arrest or Respiratory Failure. These models can be triggered with specific conditions that usually are the main factors or can also be triggered with any change of the values used by the model.

Besides the probability of the risk situation occurs, the physician can also consult the factors that contributed for that prediction. In Figure 6 is an example of a prediction of a Cardiac Arrest with 96% of probability. The physician can look at the chart and see that the negative fractions are values that are normal, or don't contribute for the condition of Cardiac Arrest and the positive ones are the ones that contribute to the cardiac arrest.

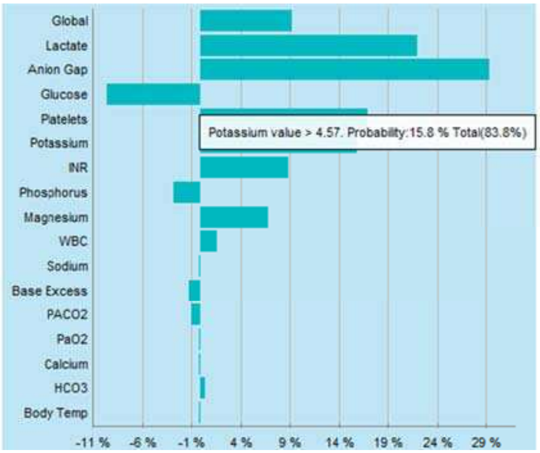


Fig. 6 - Prediction explanation

5 Proof of Concept and Demo

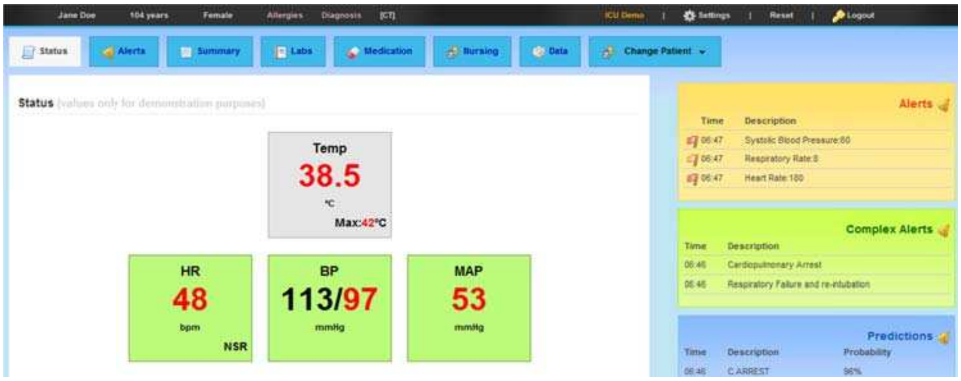
The main goals of this prototype from a medical point of view are: customizable alerts for each patient, reducing alert fatigue, predictive alerting and data mining prediction and evolution. The application developed allows to:

- Monitor the patient vitals, labs and information;
- Easily define custom ranges for alerts for any combination of patient, patient-doctor, or doctor;
- Query patient information history;
- Reduce alert fatigue;
- Support predictions using data mining models.

5.1 Development

The development of this prototype took 6 months from preparing the use case, through the design and implementation of the architecture, definition of the structure and flow of rules manager classes and rules, and development of the web interface Figure 7.

After the understanding of the architecture, the implementation reaches the major complexity implementing the support for the use case inside RM. The definition of consumption rules, callback functions for rules, cascading rules through generation of control events.



Status (values only for demonstration purposes)

Temp 38.5 °C Max:42°C

HR 48 bpm NSR

BP 113/97 mmHg

MAP 53 mmHg

Alerts

Time	Description
06:47	Systolic Blood Pressure:60
06:47	Respiratory Rate:8
06:47	Heart Rate:180

Complex Alerts

Time	Description
06:46	Cardiopulmonary Arrest
06:46	Respiratory Failure and re-intubation

Predictions

Time	Description	Probability
06:46	C.ARREST	96%

Fig. 7 - Interface

5.2 Demonstration

In this demo the main features of the system will be demonstrated. The users can update data, watch the alerts being triggered depending on the patient or doctor, define specific rules, being alerted from data mining models when high risk situations may occur. The use case is discussed elsewhere [5].

Acknowledgement: This work was supported to a great extent by Oracle staff: Aravind Yalamanchi, Pablo Tamaya, Srinivas Vemuri and Venkatesh Radhakrishnan. Also Dr. Sean Mulvihill, Dr. Edward Kimball, and Dr. Jeffrey Lin at University of Utah Health Sciences Center.

6 References

[1] Oracle® Database Rules Manager and Expression Filter Developer's Guide 11g Release 1 (11.1). Available at: http://download.oracle.com/docs/cd/B28359_01/appdev.111/b31088/toc.htm, accessed May 2009.

[2] Oracle® Database PL/SQL Packages and Types Reference 11g Release 1 (11.1. Available at: http://download.oracle. com/docs/cd/B28359_01/appdev.111/b28419/d_cqnotif.htm, accessed May 2009.

[3] Oracle® Database Advanced Application Developer's Guide 11g Release 1 (11.1). Available at: http://download.oracle. com/docs/cd/B28359_01/appdev.111/b28424/adfns_flashback.htm, accessed May 2009.

[4] Oracle® Database PL/SQL Packages and Types Reference 11g Release 1 (11.1). Available at: http://download.oracle. com/docs/cd/B28359_01/appdev.111/b28419/d_datmin.htm, accessed May 2009.

[5] P. Bizarro, D. Gawlick, T. Paulus, H. Reed, M. Cooper. Event Processing Use Cases Tutorial. DEBS’2009.

[6] D. Gawlick, Adel Ghoneimy, Zhen Hua Liu. How to Build a Modern Patient Care Application. To be published at HealthInf 2011.