Smart Patient Care

Diogo Guerra, Pedro Bizarro, and Dieter Gawlick

Abstract. The creation, management, and use of Electronic Medical Records (EMR) is a central issue for the medical community and is a high priority for many governments around the world. Collecting, storing, and managing EMR is expensive and difficult due to a set of demanding requirements for quality attributes (reliability, availability, security), multiple types of data (real-time data, historical data, medical rules, medical vocabularies), and data operations (raising alarms, pattern detection, or predictions). The traditional approach uses a combination of multiple data management systems such as databases, rule engines, data mining engines, event processing engines and more. Having multiple data management systems leads to "islands of data", missed correlations, and frequent false alarms. However, recent advances in database technology have added functionality to database systems such as temporal support, continuous queries, notifications, rules managers, event processing and data mining. This chapter describes a prototype, SICU, that using those advanced functionalities, implements a complete, single-system EMR engine to monitor patients in emergency care units. SICU was designed as a proof-of-concept EMR system that manages real-time data (vitals and laboratory data), historic data (past clinical information), medical knowledge (in the form of rules) and issues appropriate alarms with the correct level of criticality and personalized by doctor or patient. In addition, using data mining models built from real patient profiles, SICU is able to predict if patients will have a cardiac arrest in the following 24 hours. The prototype has shown a way to significantly enhance evidence based

Diogo Guerra
FeedZai, Portugal
e-mail: diogo.guerra@feedzai.com

Pedro Bizarro
University of Coimbra, Portugal
e-mail: bizarro@dei.uc.pt

Dieter Gawlick
Oracle, California
e-mail: dieter.gawlick@oracle.com

S. Helmer et al.: Reasoning in Event-Based Distributed Systems, SCI 347, pp. 217–237.
springerlink.com

medicine and is therefore of great interest to the medical community. The lessons can be applied to other domains such as smart utility grids.

1 Introduction

Collecting, storing, and managing EMR is expensive and difficult because it needs two types of data management requirements. First, as with other sensitive datasets, managing EMR requires reliability–records can never be lost, availability–records must be available at all times, and security–records can never be available to unauthorized people or systems. However, reliable, secure access to EMR records at all times is not enough [7]. Indeed, a complete EMR system should have a second set of requirements aimed to support doctors: active analysis of the many types of patient data, managing and checking medical rules, or predicting current and future patterns. While the first set of requirements is usually supported by a commercial database management system, up until now the second set of requirements was normally supported with the help of custom-code or additional data management systems such as rule engines, data mining engines, or event processing engines.

Besides being more expensive and harder to manage, having multiple data management systems frequently leads to the inability of finding correlations across datasets which in turn leads to false alarms and false negatives. However, recent advances in database research have added functionality to database systems such as temporal support, continuous queries, notifications, rules managers, event processing and data mining.

This chapter describes a prototype, SICU, that using multiple advanced functionalities, implements a complete, single-system EMR engine to monitor patients in emergency care units. Created by the University of Coimbra in cooperation with the University of Utah Health Science Center (UUHSC) and Oracle Corporation, SICU is a single integrated EMR prototype system built inside an Oracle database. SICU was designed as a proof-of-concept EMR system that manages real-time data (vitals and laboratory data), historic data (past clinical information), medical knowledge (in the form of rules) and issues appropriate alarms with the correct level of criticality and personalized by doctor or patient. Using data mining models built with 725 real patient profiles collected over 4 years, SICU aims to predict if patients will have a cardiac arrest in the following 24 hours

The prototype focuses on new ways to represent data (both patient data as well as medical rules), extraction of evidence from available data (using rules, classifications, and models), and on allowing customized interpretations over the same data. These characteristics of the prototype were implemented with an architecture based on the Oracle Database and focused on four of its features:

- *Total Recall (TR)* – a recent feature of Oracle database, responsible for automatically managing the history of all data changes [16];
- *Continuous Query Notification (CQN)* – a feature responsible for detecting and announcing changes in data [2];

- *Business Rules Manager (BRM)* – a new complex event processing (CEP) engine inside the database which is responsible for classification and personalization of alerts [1];
- *Oracle Data Mining (ODM)* – responsible for detecting complex patterns and identifying new classes in the data. It has been a feature of the Oracle Database for many releases [4].

Figure 1 shows the flow of data and the organization of the different modules in the system.

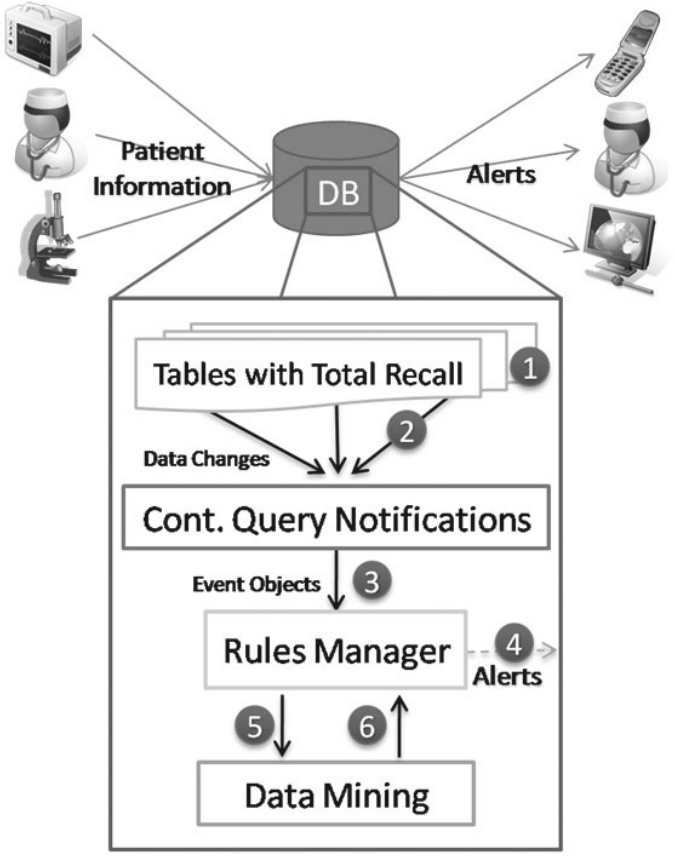

Fig. 1 Architecture of the prototype

1.1 Contributions

The major contributions of SICU are:

- *Automatic data history* – Using Total Recall (Section 2.1.1), past clinical data is automatically and transparently managed by the system and is

accessed using minimal SQL enhancements. Then, using Continuous Query Notification (Section 2.1.2), the historic data is used to detect significant changes (or lack thereof) between any two or more versions of a value.

- *Canonical event type model* – In spite of the multiple data types (e.g., heart rate, blood pressure, temperature, urine information) and multiple rule types, the implementation reduced all new data to a single type of event (a "new reading") and to a single type of rule greatly simplifying the architecture. (See Section 4.1.)
- *Representation of vocabularies and domain knowledge* – Vocabularies represent the semantics of data specific to the domain. SICU stores the terms and concepts of the domain knowledge using rules that conform to generally accepted medical rules. (See Section 4.2.)
- *Classification and Customization framework* – Although there are general rules that always apply, each doctor has her own interpretation of the importance of events (e.g., is a heart rate of 100 beats per minute too much for some patient?) and the frequency and urgency of warnings. SICU allows the customization of thresholds, alarms, timeouts and the classification of events into multiple classes according to their criticality such as guarded, serious and critical. (See Section 4.3.)
- *Rules Composability* – The complex scenarios seen in real medical systems are sometimes too complex to be defined using a single rule. As such, the prototype builds upon a rule composability property that allows the definition of complex rules to represent events such as "possible cardiac arrest situations". (See Section 4.4.)
- *Predictive Models and Data Mining Integration* – Although there is plenty of medical knowledge that can be captured using simple and composable rules, there are situations where a predictive model is able to capture knowledge that not even the specialists knew about. SICU uses a data mining model that, using 725 real patient profiles collected over 4 years, is able to predict with 67% accuracy if patients will have a cardiac arrest in the following 24 hours. (See Section 4.5.)
- *Declarative Applications* – This prototype illustrates a new style of rule driven applications that provides timely access to critical information and that can handle enormous complexity while remaining highly customizable and extensible and while using a declarative approach. (Described better in another publication [11].)

In addition to the above major contributions, the prototype has shown the importance of database technology to event processing. In fact, by taking advantage of the declarative nature of database technology, and in spite of using so many new features and being implemented by a four-site team (in Utah, California, Massachusetts, and Portugal), the prototype was implemented in just 4 months. In the process, the implementation of the prototype has also shed new light on techniques for evidence discovery.

1.2 Chapter Outline

Although we assume that readers are generally familiar with database technology, Section 2 highlights important functional and operational characteristics of the Oracle database. Then, and in more detail, Section 2 describes some of the more modern features of databases that readers may not be familiar with. Additionally, section 2 describes and defines *classification* and *customization*.

Section 3 describes the overall architecture of the prototype. The approach of this section tries to explain to the reader the architectural decisions made along with the prototype development and how they were executed. The section will highlight the way that the components described in Section 2 were linked and integrated in a unified system.

Section 4 describes in detail the implementation, how data are captured with Total Recall, how events are triggered with Continuous Query Notification, how events are further processed by the Business Rules Manager, and how the Business Rules Manager is also used to customize and compose rules. The last subject is the integration of Oracle Data Mining, the creation of models as well as the on-line scoring of incoming patient data to compute the probability that the patient will have a cardiac arrest.

Section 5 describes alternative implementation approaches. Section 6 gives an overview of the state of the prototype, Section 7 give our conclusions and highlights future work.

2 The Technology

The prototype was built on top of the Oracle database and of the Glassfish [9] application server. This section describes features from the Oracle database that are important to the prototype and are not well known in the community.

2.1 Oracle Database

Databases are known for their operational characteristics, and high performance, scalability, security, and availability, including disaster tolerance. Any of these characteristics are required for the management of EMR (Electronic Medical Records).

In addition, databases have to support the appropriate data types and models for each specific application. Oracle database initially supported only SQL 92 [12], a simple relational model. This support has been dramatically extended. As of now, the Oracle database supports SQL 99 [13], XML [14], and RDF [15]. Furthermore, the database provides support for extensibility; e.g., users can add new data structures as well as operators to manage and access data represented by these data structures. Oracle has used this extendibility support to support text processing, video, spatial, as well as other data types. In the medical environment the extensibility can be used to manage and access domain specific data such as MRIs (magnetic

resonance imaging) or to extend the richness of existing data types to support classification and customization (which we discuss further in Section 4.3.)

Keeping up with the evolution on data types and models, the Oracle database now also provides a full array of data management services. That is, the database is not just able to manage and store transactional data, but can also support the (online) maintenance of warehouses, automatically manage and provide access to the history of records (with Total Recall-TR), support fast complex event processing based on registered queries (with Continuous Query Notifications-CQN), support storing, managing and evaluating business rules (with the Business Rules Manager-BRM- module), support the development, testing and scoring of non-hypothesis driven and/or predictive models (with Oracle Data Mining-ODM), and supports the dissemination of information through queuing mechanisms.

The following section will introduce TR, CQN, BRM, and ODM briefly.

2.1.1 Total Recall

Total Recall [16] is a database feature to automatically manage the database history. Whenever a record is changed or deleted the previously existing version will be transparently stored into a history table. Automatic management of database history represents significant savings in application development because programmers can have the benefits of accessing any past versions without worrying with the details on how to store, index, or access those versions.

Total Recall uses the application schema (vocabularies) and extends it with special time fields. Thus, each record has a start time, the time of the creation of the version, and an end time, the time a version has been replaced by a newer version or the record has been deleted. The end time for currently valid records is infinite or undefined.

Users can access previous records by using so called *flashback queries*, that is, standard SQL queries extended by an optional AS OF *timestamp* clause (see Figure 2). The execution of any query decorated by AS OF will access the versions that were valid at the specified timestamp. This allows reviewing the status of a patient as of a specific time in the past. Other sophisticated expressions can evaluate the changes in a field between two points in time. The default for AS OF is the current time, e.g., programs that are unaware or do not use this feature will work as if Total Recall does not exist.

The second language extension for Total Recall is a VERSIONS clause with two time parameters representing a time interval. VERSIONS provides access to time series. A version query asking for a specific record between t1 and t2 will return all versions of this record that existed during this interval (see Figure 2).

Total Recall preserves the performance of the active data by separating the currently active versions from previous versions.

Efficient storage management is especially important for historic data. Total Recall uses compression technologies to minimize storage consumption; e.g., it will use compression algorithms that are optimized for read only. It also takes

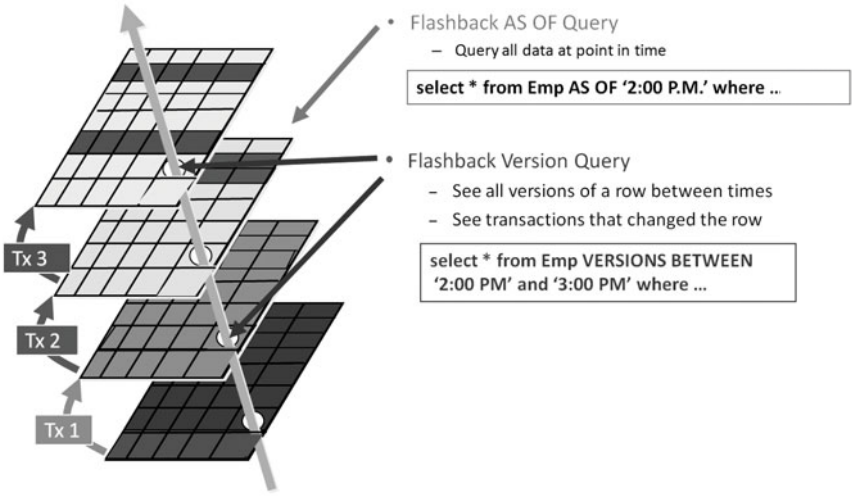

Fig. 2 Oracle Total Recall

advantages of the high compression rates that are common in columnar stores by treating several versions of a field as a column.

2.1.2 CQN - Continuous Query Notification

Many applications have requirements for taking actions when something happens in the database such as when a record is changed, deleted or added. Usually this type of functionality can be achieved with the use of database triggers.

An alternative to triggers is Oracle CQN. Oracle CQN allows an application to register queries with the database for either object change notification or query result change notification. An object referenced by a registered query (e.g., a table) is a registered object.

If a query is registered for object change notification (OCN), the database notifies the application whenever a transaction potentially changes an object that the query references and commits those changes, whether or not the query result changed. On the other hand if a query is registered for query result change notification (QRCN), the database notifies the application only for committed transactions that indeed changed the result of the query.

CQN complements existing methods of event discovery in the database. Unlike triggers that see *dirty data* (i.e., uncommitted data) and may have issues scaling with the number of conditions, CQN sees only committed data and scales better. Log mining is another standard event discovery technology: it is, however, typically optimized for batch processing and the evaluation of single elements and is not considered further here.

2.1.3 Rules Manager

The Oracle Business Rules Manager (BRM) provides a set of tools and techniques to define, manage, and enforce business rules from core business components. The Rules Manager enables better integration of business components such as CRM, ERP, and Call Center and enables automated workflows [1].

BRM also introduces PL/SQL APIs to manage rules in an Oracle database. It also defines an XML and SQL based expression algebra to support complex rule applications. The business Rule Manager leverages the Expression data type and the Expression Filter index [1] to store and evaluate the rule conditions efficiently.

BRM allows application developers to define groups of rules that are related; e.g., all rules related to cardiac arrest. Policies can define which result will have preference if there are conflicting results.

BRM also allows the integration of external data (events) with OCN events; e.g., look at room conditions in accessing a patient's situation, assuming room conditions are external data.

2.1.4 Oracle Data Mining

Oracle Data Mining (ODM) implements a variety of data mining algorithms inside the Oracle database. These implementations are integrated right into the Oracle database kernel and operate natively on data stored in the relational database tables. This integration eliminates the need for extraction or transfer of data between the database and other standalone mining or analytic servers.

The relational database platform is leveraged to manage models and efficiently execute SQL queries on large volumes of data. The system is organized around a few generic operations providing a general unified interface for data mining functions. These operations include functions to create, apply, test, and manipulate data mining models. Models are created and stored as database objects, and their management is done within the database - similar to tables, views, indexes and other database objects.

In data mining, the process of using a model to classify, or "score", existing data is called "scoring". Scoring is traditionally used to derive predictions or descriptions of behavior that is yet to occur.

Traditional analytic workbenches use two engines: a relational database engine and an analytic engine. In those setups, a model built in the analytic engine has to be deployed in a mission-critical relational engine to score new data, or the data must be moved from the relational engine into the analytical engine. ODM simplifies model deployment by offering Oracle SQL functions to score data stored right in the database. This way, the user/application developer can leverage the full power of Oracle SQL (e.g., in terms of parallelizing and partitioning data access for performance).

ODM offers a choice of well known machine learning approaches such as Decision Trees, Naive Bayes, Support vector machines, Generalized Linear Model

(GLM) for predictive mining, and Association rules, K-means and Orthogonal Partitioning Clustering and Non-negative matrix factorization for descriptive mining. Most Oracle Data Mining functions also allow text mining by accepting Text (unstructured data) attributes as input.

2.1.5 Operational Characteristics

Databases are known for their rich support of operational characteristics. These characteristics are available to applications without any additional development, and indeed operational characteristics can significantly reduce the development effort, such as using Total Recall to provide automatic and complete auditing and tracking. This section highlights some of the major operational characteristics that are important to the prototype.

Performance. Databases can store tens of thousands of records per second. The response times for storing and retrieving records and for evaluating a large number of registered queries are well below human awareness.

Scalability. Databases can store very large amounts of data, handle large amount of clients concurrently, and evaluate tens of thousands of registered queries. Compression and other technologies are used to minimize storage requirements.

Availability. Databases are known for not losing records. Recovery, restart, fault tolerance for continuous availability, as well as disaster tolerance are all standard features.

Security. Databases provide many security-related features, such as access control, fine grain security - e.g., a doctor can only see patients that are assigned to him/her, contextual security - e.g., a doctor can see only data when she is in the hospital. Additionally, all data items as well as all journals can be encrypted in a way such that not even IT personnel with unlimited access to the metadata will be able to decode the encrypted data.

Auditing and tracking. Journals provide a full account of any activity in the database; e.g., there is a full record of who did what at what time. Total Recall and a feature called Audit Vault in conjunction with Total Recall provide online access to any historic data as well as to the information of who did an action at what time.

2.2 Other Technologies

During the design of the prototype, we learned that doctors wanted to be informed of situations such as “patient has a critical temperature for more than three minutes”. This type of semantic interpretation of observed data, i.e., “critical temperature”, poses two challenges: classification and customization. *Classification* consists of assigning a class (e.g., normal, guarded, serious, or critical) to a range of observed

values. *Customization* consists of having two or more health care professionals with potentially different classes or potential different ranges of the same class. E.g., for one doctor, a temperature above 40 C may be considered "critical", while for another doctor temperatures up to 41 C should be only "serious". The next two subsections further clarify these two concepts and describe how the prototype incorporated them.

2.2.1 Classification

When people interpret data, they tend to associate individual values or intervals to some classification. For example, doctors frequently use terms such as serious and critical to identify values of concern or for communication between colleagues. Decision tables can be used to define the mapping between values and classes for a given set of attributes. In some cases, such as an EKG (electrocardiogram), there are already classifications. In this case one can map the EKG specific classifications to more generic classifications.

Using classifications simplifies and generalizes the formulation of queries and rules. With classification one can ask for all patients with at least one critical value or for the immediate notification when a patient has at least one critical value as well as a periodic reminder. Note that using classification has the added benefit of reducing the number of rules needed to monitor the systems. That is, a single rule, e.g., "alert when a parameter is critical", can be used to monitor hundreds of different parameters.

Last but not least the model can be used to classify the urgency of the information. If a rule is marked either as serious or as critical doctors will be alerted right away. The difference between critical and serious will tell them how fast they have to react. If a rule is marked guarded, doctors will not be immediately alerted at all, however, an entry will be added to the patient records and they will see the information when they are looking at these records the next time. Obviously, this will reduce unnecessary interruption and most importantly, it reduces alert fatigue.

The prototype has shown that the use of classification fundamentally reduces the complexity of the acquisition of data; classification is an important part of the metadata. Storing these metadata with TR (Total Recall) provides a full history to the classification; e.g., it becomes clear what the classification was at any point in time. The prototype supports only a specific classification: normal, guarded, serious, and critical. In general one would allow users to specify one or more classifications.

In some cases, classification was performed taking the specific situation of patients into account. For example, a heartbeat of 150 can be classified as normal or guarded for a baby but as serious or critical for a 90+ year old patient.

Classification could be extended to time series. For example, classes could be assigned to deteriorating (i.e., monotonically decreasing) attributes or classes could be assigned depending on the speed of deterioration; e.g., a value is slowly/rapidly deteriorating. These classification techniques were not used in the prototype.

2.2.2 Customization

The use of classifications simplifies significantly the interpretation of data and the specification of very generic queries and rules for ad-hoc information as well as for event processing. However, not all institutions and doctors will classify the same information in the same way. The solution to this problem is customization of the classification.

The prototype provides generic classification; e.g., classification that applies if there is no overriding customization. There are four levels of customization: generic, protocol, doctor, doctor/patient

- *Generic* – a classification used for any patient, regardless of protocol or doctor;
- *Protocol* – a classification that has been adjusted to the treatment plan that is used for patients of a specific protocol;
- *Doctor* – a classification that reflects the preferences of a doctor independent of the patient. It reflects the personal view of a doctor in respect to the interpretation of patient data;
- *Doctor/Patient* – a classification that is tailored to the view of a specific patient by a specific doctor.

In all cases the most restrictive classification will be used; e.g., doctor/patient overrides doctor customization, which overrides protocol customization which overrides the generic customization.

Customized classification allows doctors to fine-tune systems to their needs and their view of medicine. This is a very important step in increasing the acceptance of any system as well as reducing alert fatigue.

A consequence of customization is that two doctors looking at the same data may get different results. Obviously, they may get different alerts as well. One of the major issues is the very easy and intuitive customization of values. The prototype provides this by showing the current setting for the majority of vital signs in the form

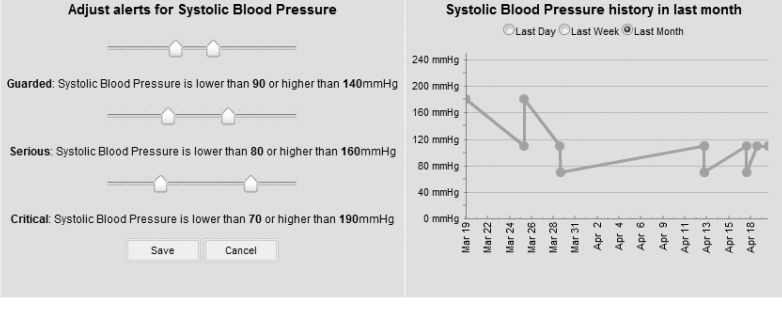

Fig. 3 Sample interface for customization

of sliders and numerical values. Doctors can easily adjust the settings by moving the sliders; the numerical values will be automatically adjusted (see left side of Figure 3). The prototype ensures that the new settings are consistent; e.g., that the values for critical are higher/lower than the settings for serious. Once a doctor saves the settings they will be used as the doctor/patient classification.

The prototype has a fixed customization hierachy (doctor/patient, doctor, protocol, generic). In a product one would allow users to define their own customization structure.

3 Architecture

In this section the high level architecture of the SICU prototype will be explained, including relevant characteristics of each module as well as some implementation decisions. As mentioned, the architecture of the prototype is database centric. This means that all the components of the system are running embedded in the database engine. This also means that operational benefits such as transactionality, logging, high availability, and high performance are implemented by the database engine and the prototype inherits them with no extra code or effort. The architecture of SICU relies only on modules of the Oracle Database. Without ignoring other important modules, and as mentioned previously, there are four main technologies of the Oracle Database that were used in the core of the system: Total Recall, Continuous Query Notification, Business Rules Manager and Oracle Data Mining.

One of the first concerns in this prototype was how to maintain a history of the data without having extra logic in the application (i.e., without using support tables or filtering timestamps). This was a decision with a potential impact on overall performance. Total Recall was chosen because it can maintain history transparently and with very good performance. Enabling this option (See step 1 in Fig. 1) will cause all operations (inserts, deletes or updates) to be recorded transparently with a timestamp enabling change tracking in time.

In a classic database, if there is a need to store current and historical values of a sensor the normal approach is to add application logic and write time-stamped records. In this traditional approach, a sensor with 100 readings takes 100 records in the table(s). However, using TR 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 second concern when designing the architecture was how to keep track of changes in sensors and ensure that the data treated was consistent (i.e., the sensor that measures the cardiac rhythm is different from the one that measures the respiratory rate. However when the system needs to analyze new data, it should see the status of all sensors at the same time). Continuous Query Notification (See steps 2 and 3 in Fig. 1) was chosen to track data changes. This technology allows the database engine to notify clients about new, changed or deleted data at commit

time. Note that CQN has some similarities with triggers. However, while triggers fire when SQL statements are executed, CQN only notifies once data is committed and consistent.

For every new new reading, the incoming data needs to be processed, classified and matched with some intelligence loaded into the system to produce valuable information to the users. Rules Manager (See step 4 in Fig. 1) is a rules engine running in the Oracle database and allows matching data against rules.

Because Rules Manager (RM) is running embedded in the database, it has access to all the information stored in it. This allows RM to take decisions considering both the new information as well as the history of that patient and/or of groups of patients that share physiological or clinical characteristics. Some of the rules designed for the prototype notify Oracle Data Mining to re-score data and detect critical situations.

When Rules Manager triggers Oracle Data Mining (See step 5 in Fig. 1), a query gathers the information needed from the sensors specified by the rules and then ODM scores the data for a probability of occurrence of a dangerous future event. The algorithm also provides information for the users to understand how that probability was calculated and the weight of each parameter. At the end of scoring, the result is sent back as a new event to RM (See step 6 in Fig. 1) where it will be analyzed.

Finally, the information is delivered to the end user through a Java EE application running on an Application Server. (See step 4 in Fig. 1.)

In short, sensors continuously send new readings to the database which are then stored with the help of Total Recall. Continuous Query Notification is always monitoring incoming data and notifies Rules Manager only of the changed values considered relevant. Rules Manager is where all the knowledge (represented as rules) resides and with those rules the incoming data is correlated (with a context) and classified, and alerts are sent to users. Some Rules Manager rules trigger the predictive Data Mining models, which score incoming data and detect possible complex scenarios such as cardiac arrests.

4 Implementation

The development phase of the prototype took about 4 months of a developer with the support of an Oracle technical team. This section describes the implementation details of the prototype.

4.1 Event Triggering

Rules Manager bases its processing on event objects, therefore it was decided for more flexibility the system would use CQN to generate those events and send them to be consumed by RM calling its *add_event* function.

The sources of information are very heterogeneous (bed side monitors, imaging, text-based doctors reports, lab results, etc) and the system can be monitoring either numeric simple values or text based reports. This leads to a problem when defining rules for extracting the desired information from the raw data because it would lead to an explosion of event data types and rule data types. (In CQN rules have data types, which must match the types of the events that may trigger the rule evaluation.) To simplify the design of the system, all incoming pieces of data from sensors were considered to be of the same datatype: a measurement (the value of the reading). All rules where considered to be rules of type array of measurements.

The two main advantages of this decision are: i) all rules are defined based on only one type, and ii) the events are smaller (if a row in the database is updated but only a column is changed then only that value generates an event).

The Continuous Query Notification is the module responsible for comparing the current and new values of the information received, generating the events and sending them to RM at commit time. This processing is defined in a callback function generated dynamically for each table that CQN is monitoring.

4.2 Rules Evaluation

Rules Manager is the main processing component of the prototype. This component provides a flexible environment to evaluate rules with several options for deciding the ordering of rule execution (when multiple rules can fire), event consumption policies or duration of events.

Ordering of rule execution is determined as follows. When an event arrives, all rules that can be triggered by the event are identified as candidate rules. Then, a conflict resolution module is executed to determine which candidate rules should be triggered and in which order they should fire. In the prototype, rules are grouped by a priority group number and are also classified by criticality level. Starting with the highest priority group with candidate rules, the most critical candidate rule is identified and fired. The process is then repeated for all other groups, ordered by priority, where for each group, only the most critical candidate rule fires. The priority of rules can be changed through a user interface.

Another configuration option implemented in the prototype was the concept of duration of alerts: when an alert is issued it has a duration period and within that period another instance of the same alert cannot be issued. This feature reduces alert fatigue. To achieve this functionality each time a rule is candidate for triggering it is also compared with the last time an instance of that rule/alert was triggered to see if the duration was expired or not.

After the full evaluation and conflict resolution the events are kept in the pool of events because they can be used by other rules in conjunction with other events. For example, the event “Temperature is above 41C” should be not consumed by the rule “Trigger serious alert if temperature is above 41C” because there might be another rule, e.g., “Trigger critical alert if bpm is above 100 and temperature is above

41C", that might need the event. However, events could not be left unconsumed in the system forever. Thus, a time-to-live or event duration was designed into the system. The final event duration was decided by the medical personnel as 24 hours. After these 24 hours in a intensive care unit, any values are considered obsolete and should be considered part of the patient history. Although event duration was set to 24 hours, RM supports other ways to specify event duration such as by call, session or transaction.

4.3 Rules Customization

Medical staff were particularly interested in a system with a high degree of customization of rules. In fact, due to the lack of personalization, doctors cannot customize the alerting modules of most other current EMR systems or can only make simple, threshold type customizations. That lack of personalization in current EMR systems in turn leads to a high number of false alerts. The customization desired by doctors led to the implementation of the following options that determine when rules fire or not:

- If the rule belongs to a group of already defined rules or if it is in a new group;
- The priority inside the group;
- If that rule will apply to all patients, only for that patient, only for the patients of the doctor that is defining the rule, or only for a specific patient under that specific doctor supervision;
- The time to live of the alerts related with that rule;
- The description of the rule;
- The definition of the rule parameters to evaluate.

All of these options are defined as metadata in the rule definition table and used by the *RM* evaluation process or in the conflict resolution logic to achieve the defined functionality. With this level of customization the system can be fully personalized by the medical staff and improve the patient care due to the optimization of doctors' and nurses' time.

4.4 Rules Composability

Rules Manager provides extensive support for the design of 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.

However by composing rules it is possible to define more complex scenarios than the ones the language allows in a simple rule. Below is a sample of a scenario in medical technical language to identify possible cardiac arrest situations. The complete rule takes more than a page to be described in technical English.

II	MAP, SBP or DPB in serious/critical range for >30min
II	MAP, SBP or DBP change from serious to critical
III	Asystole (AYST)
I	Supraventricular tachycardia (SVT)
I	Arterial Fibrillation (AFIB)
I	Arterial Flutter (FLUT)
III	Ventricular Fibrillation (VFIB)
III	GCS <= 8
II	Critical low or high temperature
I	Trauma of Cardiothoracic patient
...	... (continues) ...

To represent the rule above, the prototype uses what we call complex rules. Complex rules are rules composed of other rules. In the scenario above, the corresponding complex rule can be implemented in RM with about 50 rules. The evaluation of such complex rules happens in the following way. The roman numbers on the left column above represent different rule categories with different weights. In the above example, the complex alert should occur when one rule of category III or three rules of category II or five rules of category I fire.

4.5 Data Mining Integration

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 such as Cardiac Arrest

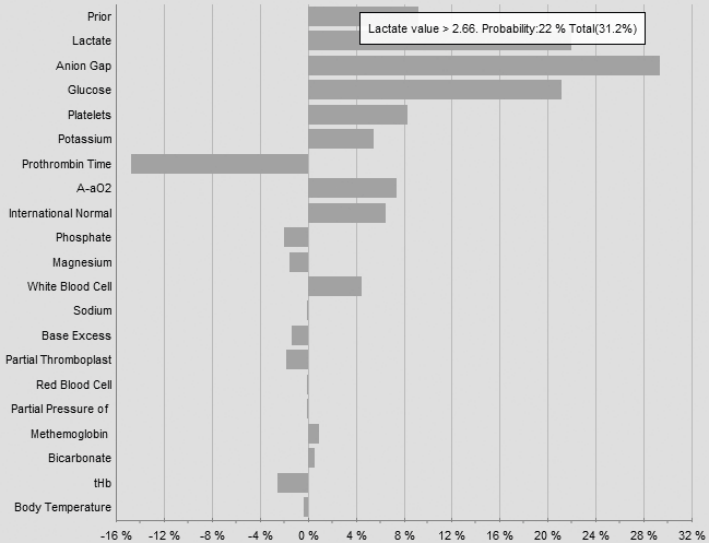

Fig. 4 Prediction Explanation

or Respiratory Failure. Since scoring the status of patients on every data change is too expensive, the process is done in two steps. First, an inexpensive rule filters the majority of non-relevant data changes. However, in some cases, the rule will trigger the execution of the more expensive scoring using the data mining models.

Scoring returns the probability of those risk situations and returns also the factors that contributed for that prediction. In Figure 4 is an example of a prediction of a Cardiac Arrest with 96% probability. The physician can look at the chart and see that the negative fractions (to the left) are values that are normal, or that do not contribute to the condition of Cardiac Arrest and the positive ones are the ones that contribute to the cardiac arrest.

5 Alternatives

The classical approach for applying event processing in an ICU (Intensive Care Unit) is to focus on vitals; e.g., blood pressure, heart beat, EKG (electrocardiogram). These vitals are shown on monitors in the patients' room and supervised in a central location. Simple limits are set for individual attributes and whenever there is an out of range value an audio alarm – an alert – is triggered. However, frequently there are too many false and simple alarms. This leads to alert fatigue and consequently even true alerts are mostly ignored.

ICUs are a favorite use case of the Event Processing community [10]. The typical approach is to capture data from the monitors and import them as streams into a CEP environment. The measurements are considered as being events. Rules and (in advanced environments) models are used to evaluate or score the incoming data to create (complex) events, which will be brought to the attention of the medical personal. In some cases, information will be additionally stored in a data base. Because of its many flaws, we dismissed this approach for our prototype. Here are some of those flaws:

- The medical personnel has to work with two inconsistent data models, one using measurements as events, one using measurements as (historical) data.
- Alerts may be created before data have been committed; e.g., auditing and tracking may not work in the presence of failures.
- Alerts are often only meaningful when new information is considered in the context of other patient data - doing so would create significant performance issues since it requires pull technology on the patient data which does not work well with the push-based, stream oriented nature of most CEP tools.
- Predictive models which are limited to vitals have in most cases marginal value; e.g., the predictive model for a 24 hour prediction of cardiac arrest showed no correlation to vitals.
- There is no classification and customization in CEP tools; the fundamental reason for alert fatigue has not been addressed.

- This approach has only very marginal value for the ICU environment while the approach taken by our prototype can be applied to any patient care.

6 State of Work

The current implementation of the application is running only for demonstration purposes, although there is an interface for the users to input and simulate data as it would occur in a real scenario and then see how the system reacts.

The main features of the implemented application are: 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 and support predictions using data mining models.

Currently the prototype has about 200 simple rules and 190 classification rules deployed. For the complex rules definition there are 160 additional deployed partial rules to trigger possible Cardiopulmonary Arrest and Respiratory Failure situations. All the rules were defined by doctors, therefore are triggering real danger situations and not only test situations.

As for the data mining integration, the prototype used a model developed by Dr. Pablo Tamayo from the Whitehead Institute/MIT Center for Genome Research, in Cambridge, Massachusetts, EUA. This model was built with a Bayesian Network using a training set of 725 patients treated at the Surgical Intensive Care Unit of the *University of Utah Health Sciences Center* in the 2004-2008 period. The model was then tested using a test set collected from 482 different patients. The results shown to predict Cardiac Arrests within 24 hours with a certainty of 67%. This represent an advance over the state-of-the-art as previsions prediction systems could only suggest a cardiac arrest with a few hours in advance [20].

7 Conclusions and Future Work

This prototype illustrates a new style of applications that provides critical information in time. These applications can handle enormous complexity, are highly customizable and extensible, and are fast to implement. In the specific implemented prototype the system first creates a knowledge base, based on vocabularies, rules (registered queries), (predictive) models, classifications, customizations, and visualization. Then, the system can improve the knowledge base by allowing users to customize or fine-tune rules and thresholds. Next, data is stored in a temporal database such that both current and past values are accessible and can be used in rules. Finally, data changes are detected and rules fire and apply predictive models as needed.

Rick Hayes-Roth coined the term VIRT (Valuable Information at the Right Time) [19]. The term VIRT seems to express the reason for and the value of event processing. The prototype shows a modern implementation of the VIRT principle.

The development of the prototype took only four months of development that focused on putting all the pieces together in one single system and delivering the functionality to the end user. This short development period shows that there are now sufficiently advanced custom-off-the-shelf components that can be put together to build sophisticated VIRT applications.

Classification is a key issue of the developed solution. Due to the prominence of classification, some more work on how to improve the ways to classify the information and how to discover new patterns to make this classification should be done. Doctors need to classify the information easily and above all the system should help and guide them to correctly classify data.

The integration of data mining models with other data acquisition and data processing systems is not trivial to achieve but has a great potential. This prototype showed that that integration is possible and can produce potential life-saving warnings.

The medical part of the team that collaborated with us showed great interest in the prototype volunteering with many extra work hours to provide feedback, improve the prototype, and demonstrate the system. In fact, Dr. Edward Kimball, from the University of Utah Health Science Center, was quoted saying "This [prototype] has the potential to revolutionize medical IT and significantly improve clinical care of a broad range of patients". Still, the system's success rests on the acceptance by the medical community in general. This means that the system needs to be easy to use and be a help to them and not something that is stopping their work. This said, the user interface to define rules and customize all the system is a part of the system that needs improvements to give this flexibility to the doctors.

One of the problems when dealing with history is the testing and load of previous history. A solution was found to move forward in the development but the system needs to have support for simulation (run in fast forward mode) and also to support load of historical data in bulk mode for loading previous data of patients when deploying the system.

Acknowledgments

The authors would like to acknowledge the great contribution of Ute Gawlick MD/PhD at the UUHSC (University of Utah Health Services Center). Ute made sure that the prototype reflected the thinking in the medical community; Ute guided the team through many challenging discussions and was instrumental in developing the ideas related to classification and customization, the visualizations, and the predictive models. We also like to thank Dr. Sean Mulvihill, Dr. Edward Kimball, and Dr. Jeffrey Lin from the UUHC for insisting on solving difficult questions and for providing suggestions solving them. Finally the authors like to thank members of the Oracle staff whos technical support was very crucial to the success of this prototype: Venkatesh Radhakrishnan, Pablo Tamayo, Srinivas Vemuri, and Aravind Yalamanch (Aravind is now working for amazon.com). Pablo developed the non hypothesis driven model for cardiac arrest, and designed the online scoring model.

References

1. Oracle® Database Rules Manager and Expression Filter Developer's Guide 11g Release 1 (11.1), http://download.oracle.com/docs/cd/B28359_01/appdev.111/b31088/toc.htm (cited May 2009)
2. Oracle® Database PL SQL Packages and Types Reference 11g Release 1 (11.1), http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28419/d_cqnotif.htm (Cited May 2009)
3. Oracle® Database Advanced Application Developer's Guide 11g Release 1 (11.1), http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28424/adfns_flashback.htm (Cited May 2009)
4. Oracle® Database PL/SQL Packages and Types Reference 11g Release 1 (11.1), http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28419/d_datmin.htm (Cited May 2009)
5. Bizarro, P., Gawlick, D., Paulus, T., Reed, H., Cooper., M.: Event Processing Use Cases Tutorial. In: The Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, DEBS 2009, Nashville, Tennessee, USA, (July 6-9, 2009)
6. Guerra, D., Gawlick, U., Bizarro, P.: An integrated data management approach to manage health care data. In: The Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, DEBS 2009, Nashville, Tennessee, USA (July 6-9, 2009)
7. DesRoches, C.M., et al.: Electronic Health Records in Ambulatory Care - A National Survey of Physicians. The New England Journal of Medicine 359, 50–60, http://content.nejm.org/cgi/content/full/NEJMsa0802005 (cited July 2010)
8. Steinbrook, R.: Health Care and the American Recovery and Reinvestment Act. The New England Journal of Medicine 360, 1057–1060, http://content.nejm.org/cgi/content/full/NEJMp0900665 (cited July 2010)
9. Glassfish Application Server, https://glassfish.dev.java.net/(cited June 2010)
10. Internet Evolution. Exploring IBM Research Labs (Part 1) - Health care analytics (January 23, 2010), http://www.internetevolution.com/document.asp?doc_id=187007&f_src=ieupdate (cited June 2010)
11. Gawlick, D.: Healthcare beyond record keeping. In: 13th International Workshop on High Performance Transaction Systems. HPTS 2009, Pacific Grove, CA, USA (October 25-28, 2009), http://www.hpts.ws/session11/gawlick.pdf (cited July 2010)
12. Wikipedia: SQL-92, http://en.wikipedia.org/wiki/SQL92 (cited April 2010)
13. Wikipedia: SQL:1999, http://en.wikipedia.org/wiki/SQL:1999 (cited April 2010)
14. Wikipedia: XML, http://en.wikipedia.org/wiki/Xml (cited April 2010)
15. Wikipedia: Resource Description Framework, http://en.wikipedia.org/wiki/Resource_Description_Framework (cited April 2010)
16. Oracle Total Recall, http://www.oracle.com/us/products/database/options/total-recall/index.htm (cited April 2010)

17. Hayes-Roth, F.: Valued-Information at the Right Time: Why less volume is more value in hastily formed networks. NPS Cebrowski Institute (2006), http://faculty.nps.edu/fahayesr/virt.html (cited April 2010)
18. Etzion, O., Niblett, P.: Event Processing in Action. Manning Publishing Company (July 2010)
19. Wikipedia: Rick Hayes-Roth, http://en.wikipedia.org/wiki/Rick_Hayes-Roth (cited April 2010)
20. Gawlick, U.: A Novel Approach to ICU Surveillance and Prediction of Disease. Presentation given on June 19, 2009 at the University of Utah Health Sciences Center (2009)