Event Cloud - Searching for Correlated Business Events

Szabolcs Rozsnyai
Secure Business Austria Competence Center
rozsnyai@securityresearch.at

Roland Vecera
Senactive IT Dienstleistungs GmbH
roland.vecera@senactive.com

Josef Schiefer, Alexander Schatten
Institute for Software Technology and Interactive Systems
Vienna University of Technology, Austria
{js, alexander.schatten}@ifs.tuwien.ac.at

Abstract

Market players that can respond to critical business events faster than their competitors will end up as winners in the fast moving economy. Event-based systems have been developed and used to implement networked and adaptive business environments based on loosely coupled systems. In this paper, we introduce Event Cloud, a system that allows searching for business events in a variety of contexts that also take the relationships between events into consideration. Event Cloud supports knowledge workers in their daily operations in order to perform investigations and analyses based on historical events. It enables users to search in large sets of historical events which are correlated and indexed in a data staging process with an easy-to-use search interface. For improving the search results, we propose an index based ranking system. We present an architecture for the Event Cloud system, which supports a continuous near real-time integration of business events with the aim of decreasing the time it takes to make them available for searching purposes. We have fully implemented the proposed architecture and discuss implementation details.

1. Introduction

Nowadays, business processes are evolving to networked workflows that are complex and executed in parallel with little human involvement to meet the needs of today's agile and adaptive business [12]. The pillars of business models are loosely coupled, distributed and service-oriented systems that generate huge amounts of events at various granularity levels.

One of the major questions is how to retrieve relevant business process knowledge out of the events in order to gain insight and knowledge about the business on an operational level for decision making.

Process knowledge is implicitly available produced by various IT systems and flow across networks in form of events. The lack of tracking those events and maintaining the causal relationships and traceability between those events, as well as aggregating them to high level events or correlating them, is a problem that is currently investigated by many research groups [4][11][13].

Traditional data warehousing approaches do a good job as a decision-making tool at a strategic level and for understanding the business situation based on a collection of historical data [8]. Although analyses on historical data, using OLAP for instance, are a way to gain deeper insights, data warehouses are not meant to provide process knowledge in the first place and regularly they often don't provide data in a real time fashion.

One of the most promising concepts that approaches the problems of gaining real-time business knowledge enable closed loop decision-making on operative levels and delivering real-time information on processes is Complex Event Processing. The term of Complex Event Processing (CEP) was first introduced by David Luckham in [11] and defines a set of technologies to process large amounts of events, utilizing them to monitor, steer and optimize the business in real time. The main application field for CEP is generally in areas where someone needs low latency times in decision cycles [9] combined with a high event throughput for observing relevant business events of predefined or exceptional situations, indicating opportunities or problems. Typically, these are areas like financial market analysis, trading, security, fraud detection, logistics like tracking shipments, compliance checks, customer care and relationship management and, in general, the monitoring of business processes and the reaction to them with a low delay.

CEP takes events with different sources, different time order and with different relationships into account. A CEP

system continuously processes and integrates the data included in events without any batch processes for extracting and loading data from different sources and storing it to a data warehouse for further processing or analysis.

In terms of business intelligence, active or zero latency data warehouses can significantly reduce the refresh cycles, but the missing process knowledge that is implicitly provided by events is still absent. The main gap in CEP solutions is the delay in analysis and thus, in taking actions, which can result in a loss of business value [9].

Event Cloud enables users to search for business events and patterns of business events within a repository for historical events. We consider this repository as a "cloud of events", which is used for searching and analysis purposes.

Event Cloud processes events, thereby creating an index for events and correlations between events in order to enable an effective event search. It provides a historic view of events with "drill-down" capabilities to explore and discover different aspects of business processes based on event correlations. Event Cloud allows users to investigate events, such as picking up single events and displaying their content and discovering related events or event patterns.

1.1. Event Cloud Example

In the following, we show an example for illustrating a typical use case for Event Cloud. In our example, we want to monitor the shipment and transports for a company specializing in the delivery of medical goods. In the field of medical logistics, it is vital to not break the cold chain of pharmaceuticals or to overrun the expiry date. During transportation and temporary storage, pharmaceuticals can be exposed to heavy surrounding conditions that can damage the goods.

Events involved in the delivery processes, starting from a demand for a specific product to the delivery activities themselves, are collected, processed and correlated by Event Cloud. Figure 1 represents such a "cloud of events".

Now, imagine a case where a user has to investigate the circumstances why the delivery of the product Tonsalumn failed on the 16.10.2006. The user would have to place a query like "Jane Smith +Tonsalumn 16.10.2006 +Vienna +Paris +FAILED" to retrieve all events and event correlations relating to that delivery process. The user retrieves a set of related events that reflect the delivery process chain (order, shipment preparation, temperature sensor data, invoices, shipment partner information) and would highlight that the transport failed because the temperature in the cooling trucks of the shipper "Mighty Trucks" exceeded the limits and so the goods were spoiled (see Figure 1, Point 1 for illustration). To dig deeper the user enters another query to retrieve information about failed transports from the previous shipper: "FAILED Mighty Trucks +Tonsalumn". The

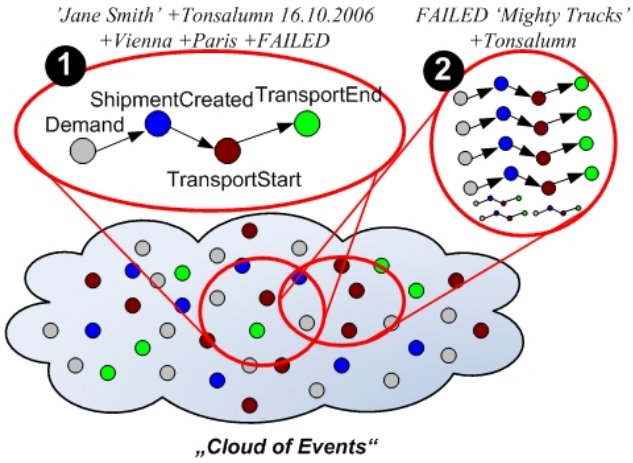

Figure 1. Extracting related business events from the cloud of events

results (Figure 1, Point 2 shows the extracted events according to the query) reveal that 9 out of 10 transports failed with "Mighty Trucks" because of temperature violations. Measures have to be taken in order to prevent further transports with "Mighty Trucks". The user creates another query to find out if there are other transports with that carrier. A query reveals that "Mighty Truck" has docked 5 minutes ago and is about to load another palette of Tonsalumn for a shipment from Vienna to Paris. At this point, the decision maker is able to react in a fast manner to avoid further losses due to an unreliable carrier.

2. Related work and contribution

Recently, event stream processing has received widespread attention. Many research projects developed architectures for querying event streams in order to discover event patterns or irregularities within the events.

The STREAM project [4] is trying to build a general data processing architecture that can support the functionalities of both DBMS and a data stream management system (DSMS). The project's focus is on computing approximate results and on understanding the memory requirements of posed queries. Event correlation over streaming data is realized with continuous queries. SQL like statements are used to select, group, aggregate and process events in the stream.

The Aurora system [5] developed an architecture to process the data streams with some QoS requirements by decomposing queries into a few predefined operators. These works mainly focus on the system architecture, continuous query execution (i.e., scheduling and various non-blocking join algorithms), and QoS delivery mechanisms.

Telegraph [6] is an adaptive dataflow system, which al-

lows to route unpredictable data flows through computing resources on a network, resulting in manageable streams of useful information.

All the above-mentioned research projects use query languages to query life data streams in order to discover event pattern and respond immediately with appropriate actions. Event stream query languages usually support operators for aggregating and analyzing event data of time windows, which provide a historical perspective of events that occurred in the past. For keeping a long history of event data, current approaches try to map event data to a database table [4] where analysis tools can query the data. By using this data mapping approach, the original structure and relationships between events are usually lost during the mapping process.

Event Cloud chooses another approach for keeping a history of business events. It stores the event objects in a database and maintains indices in order to make them efficiently accessible and searchable for analysis purposes. With Event Cloud, users don't require any data mappings between events and database tables. Keeping a history of the original business events and making them searchable has some distinct advantages such as: The ability to search for events as they occurred within a business environment in order to use them for Event analysis and mining purposes [2][3].

For creating a search index for accelerating the search for historical events, we extended the correlation approach of Schiefer and McGregor described in [13] with a ranking system for supporting different scopes for event correlation. For correlating events, correlation sets are used to define tuples declaring attributes of event types which have to match when events are correlating. During the event processing, correlation sessions are used to manage events that correlate.

3. Searching in events

Events are defined as observable actions or relevant state changes in IT systems [7][10][11]. In the most common cases, events are materialized as messages (usually in XML) that are exchanged between different peers. The representation of information about the occurred activity is usually reflected through the attributes of an event.

Events within Event Cloud correspond to event type definitions [13]. An event type definition must include at least a unique name or identifier and a collection of attributes that represent the event data.

Event Cloud makes a distinction between three types of events. *Input events* are real-world events and activities of interest like event produced by a shipment process of goods. *Internal events* are used inside the CEP solution. An example would be the detection of any change in user behaviour and, thus, an event would be fired to signalize this event for an update. *Output events* represent the observations and decisions made by the CEP solution. If some abnormal situation has been identified, the CEP system would generate an alert message to inform the authorities about this occurrence. Considering an event on its own does not expose too much relevant information, as it is too fine-grained and does not deliver information about other related events. Therefore, looking at single events does not often provide enough information to answer complex questions or to calculate business metrics.

To be able to extract meaningful and interesting information out of events, Event Cloud is capable of relating (correlating) events to other events at different stages, which manifests at three different search levels (rank 1, 2 and 3).

The definition of a correlation between event types is called a correlation set. A correlation set directly correlates event types. If we put multiple correlation sets into a relationship, we call this a bridged correlation. Bridged correlations allow the indirect correlation of event types. A correlation set consists of an unique name, the event types that participate in this correlation set, and the event attributes that relate to each other.

3.1. Search ranks

Similar to common web search engines, Event Cloud uses queries to search over events and their correlations. When ranking a search result, directly correlated events get a higher relevancy than indirectly correlated events. In other words, search queries that match strongly related events are more valuable and ranked higher than queries that match events that are indirectly related.

Event Cloud has three different search scopes which are represented by rank 1, 2 and 3. The search ranks reflect the type and intensity of relationships between events. In the following, events of a shipment process will be used to illustrate the search ranks.

3.2. Rank 1 search

A rank 1 search only considers the event attributes of a single event without considering any event correlations. The user is able to issue queries for filtering events containing specific attribute values or explicitly exclude unwanted attribute values to shrink the number of found events. An example query for "Jane Smith -Paris +Madrid" would return a result set of events containing Jane Smith and events that contain Madrid as the event attribute by all means and would exclude events that contain Paris as attribute values. Event Cloud uses a powerful query language [1] for defining search expressions.

For rank 1, the most relevant and thus the highest ranked events are events that fulfill all the provided query terms.

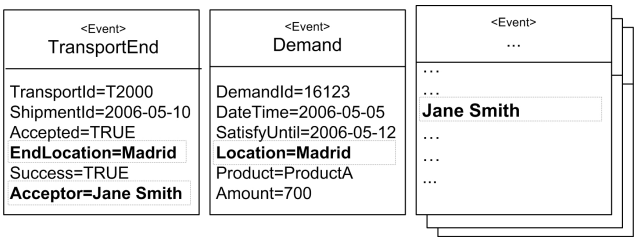

Figure 2. Rank 1 search for query "Jane Smith -Paris +Madrid"

Figure 2 shows events where a search query matched some attributes. In this case a query for *Madrid* and *Jane Smith* matched the attributes *EndLocation* and *Acceptor* in *TransportEnd*. As discussed previously, rank 1 result sets don't deliver sufficiently valuable information.

3.3. Rank 2 search

A rank 2 search extends the search scope by also considering relationships between events. If the entered search terms are found somewhere within the correlated events, these events will be returned to the result set.

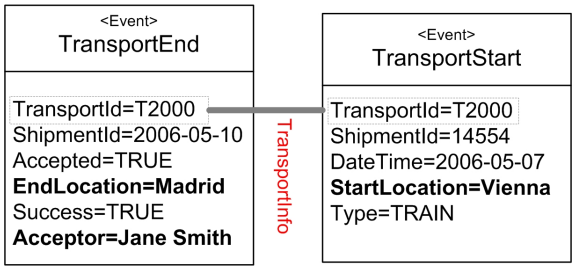

Figure 3. Rank 2 search for query "Vienna Madrid Jane Smith"

The rank 2 search can be considered as an extension of the search scope. Figure 1 shows the events represented by dots with different shades representing an event type. The lines highlight the correlations between those events. The circles indicate correlation sets. If a query returns a set of correlating events matching the queries, it is possible to navigate through those events in order to discover related events or event correlations that might be of interest.

A good illustrative example would be if the user queries for "Vienna Madrid Jane Smith" and retrieves a collection of TransportInfo correlation sets with all the events that took place in a conducted transport where *TransportStart*

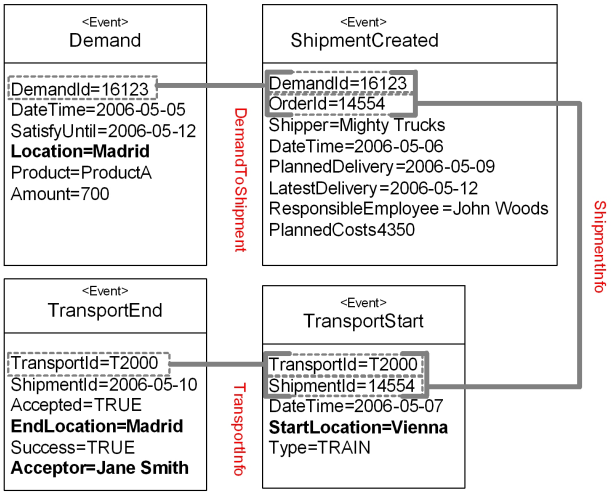

Figure 4. Rank 3 search for query "Vienna Madrid Jane Smith"

indicates that the shipment started in Vienna and the *TransportEnd* event was triggered in Madrid. In the found correlation set every involved event is available for analysis and ready to gain insights (see Figure 3). Event correlations are usually declared between event attributes and to stick to the made-up example, and it would be a transportation ID that glues the events together. A rank 1 result set contains the found events and in case of rank 2, the result set contains found correlations sets holding the corresponding events.

3.4. Rank 3 search

The rank 3 search goes one step further and represents a search between correlations. It allows a user to search for multiple terms that occur in indirectly correlated events. The result set includes all correlation sets including events that are indirectly correlated. Rank 3 searches over correlations like the rank 2 search, but instead of using direct correlations between events (TransportId in Figure 3), it searches over indirect correlations between correlation sets. The brackets in Figure 4 represent such "bridged correlations". Let's assume the user wants to examine a transport and take all related event correlations into account like the demand for a specific product that triggered the transport, the preparation of a shipment or invoicing related issue regarding the transport. A *TransportInfo* correlation would reveal information about a shipment execution. An event in that correlation would provide a bridge to invoicing the shipment preparation and then to the initial demand.

Spoken on a more abstract level, the rank 3 search expands the search space by another facet. It allows its users to search for indirectly correlated events which broadens the

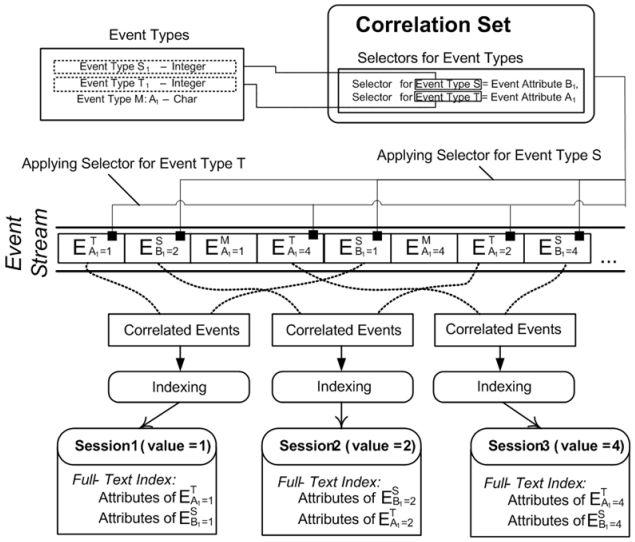

Figure 5. Correlations

space of possible matches. This larger space of possible results brings a drawback, as it might increase the number of false positives.

4. Correlating events with correlation sets

An activity spanning some significant period of time is represented by the interval between two or more events. For example, a transport might have a "transport-start" and "transport-end" event pair. Similarly, a shipment could be represented by the events "shipment-created", "shipment-delivered" and multiple transport event-pairs.

For purposes of maintaining information about business activities, events capture attributes about the context when the event occurred. Event attributes are items such as the agents, resources, and data associated with an event, the tangible result of an action (e.g., the result of a transport decision), or any other information that gives character to the specific occurrence of that type of event.

Elements of an event context can be used to define a relationship with other events. Event Cloud uses correlation sets for defining relationships between events of business activities. Figure 5 shows the application of a method of correlating events in a stream of events where E represents an event. Each event E has an identifier T, S, M etc. of the type of its event and at least one attribute A1, B1, etc. of this event. An event correlation is defined by a correlation set which consists of a set of selectors for various event types. For a given event of an event stream, an event correlation is performed successfully if 1) the event type of the given event conforms to the event type of one of the selectors of the correlation set, 2) this selector is used to extract one or more event attributes from the events, and 3) the extracted event data is used to assess the events as correlating if the attributes extracted by the responding selectors match.

Event Cloud uses defined correlation sets to collect attribute values of correlating events for business activities. The gathered attribute values are held in a data container which we define as correlation session. In other words, a correlation session is a container with a set of data items that exists for each relationship between events.

Event Cloud uses the data in a correlation session for building a Lucene [1] full-text index. Each time a new event is sent to the Event Cloud system, it automatically activates the correlation sessions for the relevant business activities and updates the full-text index. Correlation sessions enable Event Cloud to effectively gather data of correlated events which are finally full-text indexed as documents in Lucene.

5. Event Cloud Architecture

Event Cloud was designed to meet the requirements for an Event Analysis tool following David Luckham [11]: It provides access down to single events and enables the representation of event correlations, including functionalities to explore and navigate through them. The tool has a graphical interface for presenting events, their relationships and a facility to allow its users to draw conclusions. It includes a set of filters to hide and exclude unnecessary aspects and allows its users to drill down from high-level process views to low-level events.

Event Cloud runs inside an open source IoC container of the Spring Framework. Spring is used as a facility, using Spring's terminology, to glue the Event Cloud components together. The end-user interface for issuing queries and rendering the search results is implemented using HTML and AJAX and accessible through a web browser.

5.1. Preparing the search index with event services

The cornerstone of Event Cloud's architecture is the Event Server which is a multithreaded IoC container responsible for event processing by providing various services.

Events are propagated into the system through event adapters (Figure 6, Point 1). In an IT production environment, different IT systems can generate different types of messages to signalize events. Event Cloud's architecture allows the docking of a number of source systems by using adapters. In the current implementation, a JMS adapter has been used, as it is a standardized queuing system which opens access to a range of other systems by default. However, Event Cloud is able to attach other adapters to process

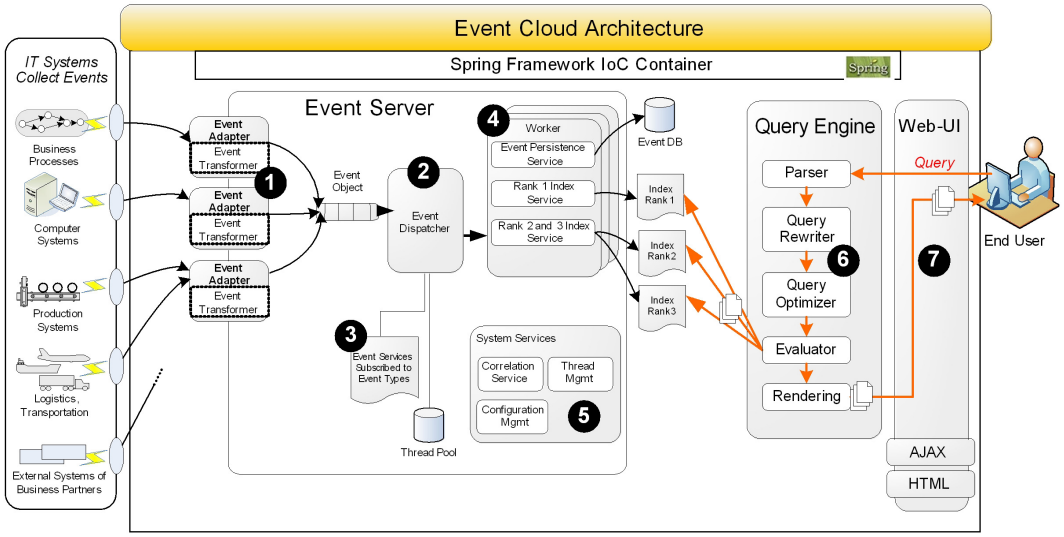

Figure 6. Event Cloud Architecture

events from different systems with different message formats.

If event data is received by an event adapter, it is passed to an event transformer, which converts the event data into a uniform event object. After the transformation is completed, the event will be placed in an internal queue for further processing.

The job of the event dispatcher (Figure 6, Point 2) is to allocate enqueued events to workers running in separate threads waiting to get an event and an event service for processing. Based on the event type definition, the dispatcher looks up the corresponding event service or the given event type and hands the service over to the event worker.

Event services (Figure 6, Point 3) implement a predefined interface and perform the processing of events. Event services are activated on request and are able to handle event correlations by retrieving the correlation state through calling the correlation service.

Event Cloud currently contains the following four event services (Figure 6, Point 4): The *Event Persistence Service* stores the events in a database. The *Index Services for the various ranks* add the given events to the corresponding full text indexes. The indexing services are used for creating the Lucene indexes for the rank 1, 2 and 3 search types.

Further extension of the system is possible through adding custom implementations of the event service interface to the IoC container.

Apart from the specific event service implementations the system contains the following system services (Figure 6, Point 5): the correlation service, thread management and configuration management.

One of Event Cloud's key services is the correlation service that is responsible for handling the creation, retrieval and the persistence of correlation sessions. Whenever an event is processed by the correlation service, either a new correlation session is created or the event is attached to an existing correlation session. Correlation sessions are used to collect the event data which is used for the full-text indexing with Lucene. For details on managing correlation sessions with a container, please refer to [13].

5.2. Full-text indexing with Lucene

The solution in Event Cloud to manage the search (rank 1, 2 and 3) and the navigation in events is to create three different full-text indexes out of the events and their correlations by treating them like documents. One full-text index holds the documents for the rank 1 search and the other index holds the information for the rank 2 and 3 searches.

A relational data model only for enabling search functionalities has proven to be a bad choice as there are several performance issues especially coming with the nature of rank 2 and rank 3 search queries. Thus, the choice was a full-text index (using Lucene) on events and their correlations. Using Lucene, event data are indexed as documents (in terms of Lucene documents are index containers storing data). Event services are used to collect relevant event data with the correlation service and build documents which are added to the full-text index.

Rank 1 index: Contains every processed event as a document (Figure 7), whereas every attribute is searchable. Further, it holds a unique ID, the event type, the event content of the event itself and a date that represents the process time. Lucene searches through the content field of a document and passes back the matching documents according to the placed query. An alternative approach is to ex-

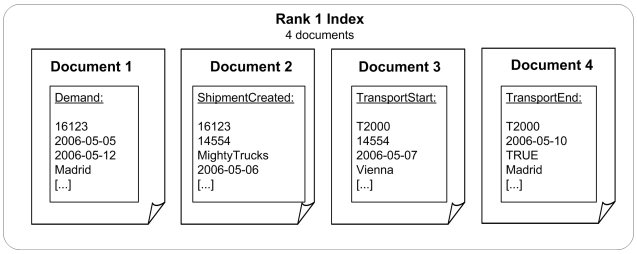

Figure 7. Rank 1 Index

clude the event data from the document and to use only a global unique ID from the found document to retrieve the event attributes from a database. Performance measures have shown that using only the index approach performed faster.

Rank 2 and 3 index: For the rank 2 index (Figure 8) events are stored with the corresponding event session ID. Rank 2 index documents contain a unique ID, the correlation name, eventguid of the event identifiers for optimization, updatedate showing the last update of the document, metrics information and dates to allow a date filter.

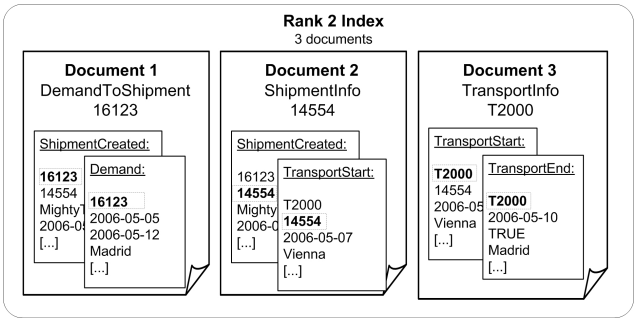

Figure 8. Rank 2 Index

Further, it contains a correlation value which stores the attribute values for the represented correlation instance (e.g. TransportID=T2000 for the correlation set TransportInfo) and the session text containing the content of every event that is in the given correlation.

The definition for the rank 2 and rank 3 indexes are equal. However, they are managed in separate indexes by separate event services. Rank 3 stores representations of bridged correlations (Figure 9), whereas rank 2 only stores direct correlations.

One of the major performance-related problems with the rank 2 and rank 3 indexing is the requirement to update existing Lucene documents. In contrast to the rank 1 where single events are indexed, rank 2 and 3 contain correlation sets. That means that every time an event comes into the system and belongs to an existing correlation instance, the corresponding Lucene document has to be updated.

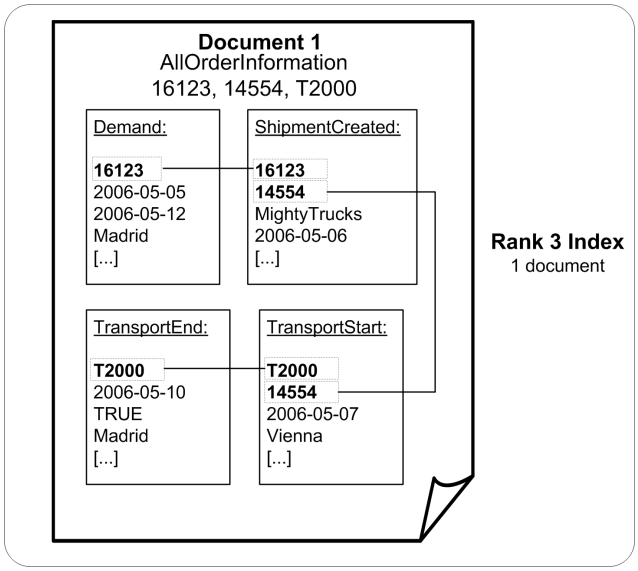

Figure 9. Rank 3 Index

6. User interface for querying events

The main entry point to find events and their correlations in Event Cloud is a simple search interface. At first glance, this interface seems to be quite simple, but it opens the possibility to place complex queries to retrieve a result set of events and their correlations for further investigation. Event Cloud provides a toolset for exploring the “cloud of events” and analyzing the relationship and dependencies of events to gain deeper insights.

The queries placed by the user are delegated to Event Clouds query engine (Lucene). Lucene is then fully responsible of parsing and processing the query to retrieve documents representing the events and their correlations from the indexes.

There are three different indexes representing the document structure of the three different search ranks. To allow Lucene to distinguish between the ranks, the user simply selects the search type in the interface through radio buttons. The selection “over bridged correlation sets” is a rank 3 search, “over correlation sets” is a rank 2 search and ”only events” triggers a simple rank 1 search.

Figure 10 shows the result set containing correlation sets with their events. On the left side, the user can open a filter toolbar for reducing the time frame of the occurred events in the result set; the user is able to exclude specific event types or correlation sets (in this case correlation sets), and he is able to create profiles which automatically only include event types and correlations into the result set that are defined by the user. The result set interface provides the opportunity to browse through the events or correlations, to drill down to single events in order to reveal their context.

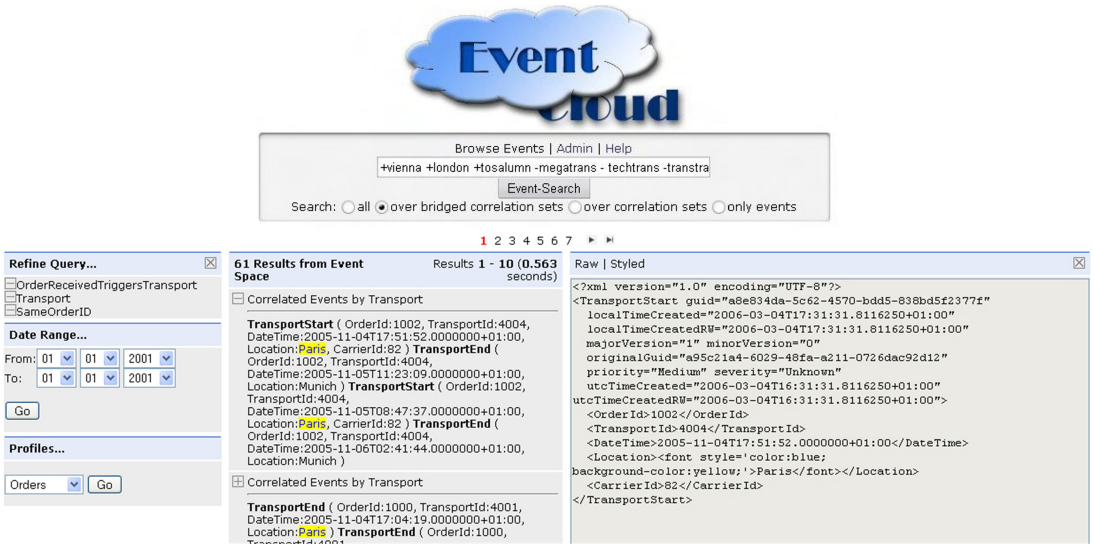

Figure 10. Event Cloud search result interface

The user interface displays the events either in a raw XML representation or in a HTML rendered version. Using AJAX allows updating the page without a full reload of the whole web page. This is especially useful to the user when applying filters or excluding unwanted events or correlation sets from the results set.

7. Conclusion

Event-based systems have been largely studied and used building and monitoring loosely coupled business solutions. This paper presented Event Cloud, an approach for flexible searching business events which have been captured by an event-based system. We showed how correlation sets can be used to model relationships between events in order to extend the scope for searching purposes. We defined three ranking levels for which we developed indexes and proposed an architecture for preparing these indexes in near real-time. The work presented in this paper is part of a larger, long-term research effort aiming at developing an event analysis toolset. This toolset will allow users to query and analyze large repositories with historical events from various sources. A key focus of this future research work will be the visualization of found events with respect to their temporal occurrence, their correlation with other events, and event clusters.

References

[1] Lucene queryparser. http://lucene.apache.org/java/, 03 2007.

[2] W. Aalst, A. J. M. M. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

[3] C. C. Aggarwal. A framework for diagnosing changes in evolving data streams. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 575–586. ACM Press, 2003.

[4] S. Babu and J. Widom. Continuous queries over data streams. *SIGMOD Rec.*, 30(3):109–120, 2001.

[5] D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. B. Zdonik. Monitoring streams - A new class of data management applications. In *VLDB*, pages 215–226. Morgan Kaufmann, 2002.

[6] A. Deshpande and J. M. Hellerstein. Lifting the burden of history from adaptive query processing. In M. A. Nascimento, M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, editors, *VLDB*, pages 948–959. Morgan Kaufmann, 2004.

[7] L. Fiege. *Visibility in Event-Based Systems*. PhD thesis, Technische Universität Darmstadt, 2005.

[8] M. Golfarelli, I. Cella, and S. Rizzi. Beyond data warehousing: what's next in business intelligence? In I.-Y. Song and K. C. Davis, editors, *DOLAP*, pages 1–6. ACM, 2004.

[9] R. Hackathorn. Current practices in active data warehousing. *DMReview*, 2002.

[10] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.

[11] D. Luckham. *The Power Of Events*. Addison Wesley, 2005.

[12] D. Luckham, A. Manens, S. Bhansali, W. Park, and S. Daswani. Modeling and causal event simulations of electronic business processes. 2005.

[13] J. Schiefer and C. McGregor. Correlating events for monitoring business events. *ICEIS 2004*, 2004.