The New BI
with Exalead's CloudView

A Whitepaper

Author:
Rick F. van der Lans
Independent Business Intelligence Analyst
R20/Consultancy

September 26, 2011

Sponsored by

Copyright © 2011 R20/Consultancy. All rights reserved. CloudView and Mashup Builder are registered trademarks or trademarks of Exalead S.A.. Trademarks of other companies referenced in this document are the sole property of their respective owners.

Table of Contents

Copyright © 2011 R20/Consultancy, All Rights Reserved.

Copyright © 2011 R20/Consultancy, All Rights Reserved.

1 Summary

The world of business intelligence is changing. The user community is demanding new forms of business intelligence applications. Together, these new forms are called ***The New BI****. Most of these new forms require technologies that have been around for some time, but have been alien for most business intelligence environments until now:* search technology *and* mashup technology*. Extending business intelligence environments with those two technologies enhances their capabilities. This whitepaper describes how Exalead's enterprise search engine called CloudView and their mashup tool called Mashup Builder allow these new business intelligence applications to be developed with which users can analyze structured and unstructured data in an integrated fashion for improving their decision-making process.*

Most people are not aware of it, but *search technology* is probably one of the most deployed computing technologies. Just consider the number of times people use web search engines, such as Google and Yahoo!. Search technology offers the functionality to search in large data sets consisting of unstructured and structured data. It can help users locate documents in which certain keywords appear. In the world of search technology, a distinction is usually made between web search engines and enterprise search engines. The latter are used for helping users find documents, texts, and data stored in their own local data stores.

Although search technology and business intelligence technology have both been around for more than twenty years, for a long time they have lived separate lives. The *business intelligence* community hasn't really adopted search technology nor has search technology seriously entered the business intelligence market. This is quite unfortunate, because combining these complimentary technologies does have a synergetic effect; 1+1 is clearly 3 here, as is shown in this whitepaper.

Several new forms of reporting and analysis can be identified where applying search technology enriches a business intelligence environment:

- *Unrestricted ad-hoc analysis;* A common need at the operational management level is to react instantly to unique problems that arise. Search technology allows managers to find a solution to those incidents for which no reports and tables have been developed beforehand.

- *360° reporting;* For certain decisions a full and extensive picture of a customer, product, or any other business object is required. This is called a 360° view or 360° reporting. Search technology allows the creation of a 360° view in which data from unstructured and structured data sources is combined, and without the need to define relationships beforehand.

- *Exploratory analysis;* For problems for which the solution is not clear upfront, search technology offers free-form querying of data. Users can freely explore and navigate the data without any restrictions (unless security-related) and no need exists for predefined reports or table structures.

Copyright © 2011 R20/Consultancy, All Rights Reserved.

- *Text-based analysis;* Search technology can be used to analyze unstructured data sources as easily as classic reporting tools can be used to analyze structured data. In addition, it can be used to derive structured data from unstructured data making reporting and analytics on unstructured data possible, and making it possible to analyze structured and unstructured data in an integrated fashion.

- *Mashup-based reporting;* Mashup technology allows for developing business intelligence applications quickly. This is a cost-effective way of developing BI applications for small groups of users or for short-lived BI applications.

This whitepaper describes these new forms of reporting and analysis, in other words it describes *The New BI*. In addition, it explains how Exalead's enterprise search engine called *CloudView* and their mashup tool called *Mashup Builder* can be used to create *The New BI*. Exalead's CloudView is an advanced, mature, and scalable enterprise search engine with which structured and unstructured data can be processed. The powerful semantic processing features make it possible to extract meaning from unstructured data. The openness of the product makes it possible to seamlessly integrate it within business intelligence systems. Exalead's history with web search-based engines, parallelization, and data slicing guarantees an efficient and highly scalable architecture. Exalead's Mashup Builder is a mashup tool for quickly developing business intelligence applications that exploit search technology.

Note: The target audience of this whitepaper consists of data warehousing and business intelligence specialists, IT managers, information managers, and IT architects. Therefore, the terminology used here is not always the standard terminology used in the search-technology community, but it's the terminology used in the data warehouse and business intelligence world.

2 The New BI – New Forms of Reporting and Analysis

Traditional Business Intelligence – The success of most organizations is highly dependent on the quality of their decision-making. Without a doubt, aspects such as the quality and price of the products and services delivered by an organization have a major impact on whether an organization is successful as well. For example, releasing a faulty car model on the market could lead to bankruptcy of the car manufacturer; or, services that are priced too high could place an organization out of business. But even if the quality is perfect and the price is acceptable, it's still not a guarantee for success. The reason is that one incorrect decision can destroy a product, even a whole organization. For example, marketing a product incorrectly or distributing products inefficiently could lead to a business disaster.

The field of *business intelligence* focuses on supporting and possibly improving the decision-making process of an organization. This is done by supplying the organization with the right data at the right time and in the right form. Note that business intelligence is not the process of making decisions, but about supporting decision-making.

Copyright © 2011 R20/Consultancy, All Rights Reserved.

Many definitions of business intelligence exist. In this whitepaper we use the one introduced by Borris Evelson of Forrester Research:

> *Business Intelligence is a set of methodologies, processes, architectures, and technologies that transform raw data into meaningful and useful information used to enable more effective strategic, tactical, and operational insights and decision-making.*

From this definition we can derive that business intelligence is not a tool nor a big database nor some design technique, but it's everything needed to transform and present the right data in such a form that it will improve the decision-making process.

Introduction of The New BI – In the beginning of business intelligence, the focus was very strongly on supporting the decision-making of the two top management levels: strategic and tactical. Very typical of this form of business intelligence is that the data used by these management levels is aggregated and the data doesn't have to be 100% up to date. Data that is one day, week, or month old, could still be useful.

But like anything else, the world of business intelligence is changing. The main reason is that organizations are changing, their way of decision-making is changing, and new technology is making new forms of reporting and analysis possible. Organizations need to focus on those forms, including operational BI, streaming BI, BigData analytics, exploratory analysis, 360° reporting, and so on. We call these new forms of reporting and analytics *The New BI*. The New BI will have a radical impact on how BI systems must be designed.

This whitepaper describes *The New BI* and five of these new forms of reporting and analysis.

Unrestricted Ad-Hoc Analysis – Besides classic forms of reporting and analytics, operational management has needs not typical for higher levels of management. In an operational environment situations might arise that require a direct response. We will use an example to describe this.

Imagine that a truck owned by a retail company was supposed to deliver fifteen pallets of a particular type of soda at a stores in Boston before opening time. Unfortunately, the truck has engine troubles and has been parked at the roadside. The challenge for the manager on duty is to find an alternative way of getting the soda to the stores. An alternative could be to send an empty truck to the stranded truck, rack the pallets, and bring them to the store. But are their empty trucks available in the area? Another alternative might be to check whether there is another store in the area from which soda can be retrieved. Or is it better to just send a new shipment? Whatever the solution, this manager needs access to up-to-date data. It would be useless to give him access to yesterday's data. Yesterday's data will not tell him where the trucks are right now.

In addition, because the solution can be anything, a manager must have access to a system that can point him to the various solutions. He should be able to freely query the available data. For example, he should be able to enter a query that includes the keywords soda, Boston, trucks, and the result should show him everything the system knows about those keywords, and hopefully somewhere in that answer he will find the best solution.

Copyright © 2011 R20/Consultancy, All Rights Reserved.

Although most data warehouse environments don't support this form of analysis, many organizations would benefit from it. Think about a hospital environment where a patient is brought in who urgently needs a particular type of operation. However, all operation rooms are occupied. What to do, find another hospital? At what time will the current operations end? Another example is an airport that has to close temporarily because of heavy fog conditions. What do you do with the airplanes that are supposed to land there? Or what do you do when the cargo doors of a recently landed airplane don't open? What do you do with the luggage? In both situations, users should be able to freely navigate all available data.

What's special about these examples is that the analysis is triggered by an incident. Something unexpected happens and the organization has to react, and it has to react fast. Classic reports work well for reoccurring problems, but not for unique cases. In a classic data warehouse environment, to let users query along non-defined relationships would involve quite a lot work. Existing tables must be extended with new data, ETL scripts must be adapted, and so on. But, because this is an incident, there is no time to do this.

In the above situation, what's needed is a new form of analysis. A form where users can freely analyze data (with almost no restrictions) without the IT department having to predefine tables and relationships. Users should also have access to up-to-date data. We call this form of analysis *unrestricted ad-hoc analysis*. The adjective 'unrestricted' is added to distinguish it from the more traditional form of ad-hoc analysis. With traditional ad-hoc analysis users can also enter any query, however, only predefined tables and relationships can be used. If certain relationships don't exist, the user can't use them for analysis. So, although the traditional form is ad-hoc analysis, it's still restricted.

360° Reporting – Imagine a call centre of an insurance company. Customers call up with questions related to their insurance. For the call centre operator it might be useful to see not only the data directly related to that insurance, but to get a *360° view* of this customer. i.e. a full picture of the customer: does he have other insurance plans, has he been sending complaints by email, did he tweet negatively about the company, does he call often and does he call about the same issue? The more data the call centre operator has available, the better his understanding of that customer will be, which could help him to support him better.

What the operator needs is a 360° view of the customer. But to create this 360° view, it's not sufficient to present data from the operational systems and/or data warehouse, because it will only contain structured data. What is needed is that data from all kinds of data sources are brought together, ranging from data coming from unstructured to structured sources and coming from internal to external sources. Being able to show a full picture of a specific concept is called *360° reporting*.

In most cases, 360° reporting is done on the lowest level of detail. It's done on an individual customer, an individual bank account, or product. Most current data warehouses have been designed to show aggregated data and primarily structured data coming from operational databases. For 360° reporting, access to the lowest level of detail is needed and access to unstructured data sources is as important as access to structured data sources.

Copyright © 2011 R20/Consultancy, All Rights Reserved.

Exploratory Analysis – Imagine a manager responsible for a group of retail stores in the Boston area who finds out that there is a problem with sales of all types of soda's in those stores: sales are not as expected. Evidently, it's important for him to find the cause. The reason might not be that evident, in fact, it could be anything. The reason could be that lately deliveries to the Boston stores have had serious delays, which means certain products are not in store on time. It could also be that the number of employees present in those stores is low due to the flu. This would probably mean that the stores operate too slowly and that could impact sales negatively. There could also be an external reason, such as customers are postponing their shopping due to bad weather conditions. Another external reason could be that a competitor has started a special promotion where sodas are sold at their stores for half the price.

Classic reporting won't help to determine the source of the problem. Reports created with those tools will show, for example, that the sales are behind, but not why. They are restricted to showing pre-defined reports, and if no report has been developed to answer this question, the manager won't find the problem.

Analytical tools offer the manager features to view data from all angles and at each possible level of detail. However, he can only discover relationships between data elements via links, relations, keys, and dimensions that have been pre-defined in the database. If, in a multi-dimensional cube, sales data is not related to delivery data or employee sickness, it won't be able to show that delivery issues are causing the problem.

Statistical and data mining tools won't help either, because these tools, although very powerful, can only create models on the data supplied. In a way, we must guide these tools. If the manager thinks delivery is causing the problem, we could use tools to determine statistically that this is indeed the case. But that's not the manager's problem, he doesn't know yet whether delivery is the problem, he is still looking for the reason. And, as indicated, the reason could be anything.

In other words, those products are very useful when we have a feeling of the reason of a problem. What this manager needs is a tool with which he can find the problem himself. When found, he can switch to the reporting and analytical tools to study the problem in more detail.

What is needed is a tool that allows the manager to query, navigate, browse, and analyze data without any restrictions. It should allow the manager to relate data elements for which no relationships or tables have been pre-defined. He should be able to ask questions such as "What's happening in Boston?" or "Anything special in the week of May 15, 2011?" And when an answer is returned, he should be able to continue along that path. This is what we call *exploratory analysis*.

The key difference between unrestricted ad-hoc analysis and exploratory analysis is that the former is used when an incident happens and a solution must be found right away. Exploratory analysis might be useful for these urgent issues, but also for those issues where a fast response is not essential.

Copyright © 2011 R20/Consultancy, All Rights Reserved.

Text-Based Analysis – On May 4, 2011, the USA Today reported that Wall Street traders mine tweets for investment clues. They monitor and decode the words, opinions, rants, and even keyboard-generated smiley faces posted on social-media sites. In other words, they're analyzing text.

Most current business intelligence systems allow users to analyze data coming from structured data sources, i.e. operational databases, but don't allow users to exploit unstructured data (text) for reporting and analysis. This is a seriously missed opportunity, because the amount of unstructured data is enormous and there is a wealth of information hidden in it. The above example of Wall Street is a perfect example of why organizations would want to analyze unstructured data (in this example tweets) for decision-making.

Many examples exist where analyzing unstructured data might improve the decision-making process of an organization. For example, an insurance company might want to analyze all contracts (textual documents) to find out how many of them will expire within one year. An electronics company might want to analyze all the messages on Twitter to find out if their products are mentioned and whether those messages are positive or not. Consider all the information in emails send out and received by an organization, all the information on the web, and in social media networks.

So the big questions are: How can these currently untapped sources of unstructured data be used for reporting and analytics? How can unstructured data be integrated with the structured data stored in current data warehouses? How can we enrich business intelligence systems to the benefit of the organization? This form of analysis in which unstructured data is analyzed is referred to as *text-based analysis* in this whitepaper.

Long-Tail Reporting – What if a business intelligence application is needed for just a few weeks, or maybe a few days? For example, a manager of a retail company is investigating whether to outsource the transport department; a decision must be made within three weeks. He needs access to particular data sources to make a solid decision and he needs a business intelligence application showing him data on the costs of insourcing and outsourcing. Whatever decision is made, after those three weeks, the application will become obsolete.

What if an application will only be used by a small number of users? For example, the CEO of a retail company uses a business intelligence application that shows him the latest sales data. Although he will probably use that application for years, he will be the only user.

For both situations, developing a business intelligence application in the traditional way is probably not worth the effort, because it would take too long or it would be too expensive. In fact, in the first situation it might take so long that when the application is ready, the need for it will have passed. That's why most of the current business intelligence applications are developed for large groups of users and/or are developed to be used for a long period of time. If the number of users is high enough or if the predicted lifespan of the application is long enough, it makes sense to spend time and money on designing and developing a robust business intelligence application.

But from the standpoint of an organization, it still might be beneficial to support this decision-making process somehow. Therefore, for certain users it must be possible to develop an

Copyright © 2011 R20/Consultancy, All Rights Reserved.

application quickly and for others it must be possible to develop applications cheaply. Note that these applications still might have complex requirements, such as integrating structured and unstructured data, accessing internal and external sources, displaying data as business graphs, and so on.

We call this form of reporting *long-tail reporting*. The term long-tail was introduced by Chris Anderson in his book 'The Long-Tail: Why the Future of Business is Selling Less of More.' In this bestselling book the author explains that it can make sense to develop products or services for small groups of customers as long as you can locate them. These markets are called long-tail markets.

Summary – In this whitepaper all these non-traditional forms of reporting and analysis are called *The New BI*. They will change the way we build business intelligence systems. They will, for example, impact the way we think about the role of the data warehouse: it won't be the only source for reporting and analytics anymore. They will require access to unstructured data sources. Making data available only from operational databases will not be sufficient anymore; users must be able to analyze structured data as well as unstructured data.

This whitepaper shows how these new forms of business intelligence can be implemented using Exalead's search and mashup technologies . Both technologies are explained and descriptions are given on how they can be embedded in business intelligence systems. Next, the search and mashup technology of Exalead, respectively CloudView and Mashup Builder, are described in detail.

3 Search Engines in a Nutshell

This section explains in a nutshell how search engines work.

The Architecture of a Search Engine – A *search engine* is capable of searching for documents, files, or values based on keywords designated as search terms. Figure 1 shows the high-level architecture of a search engine.

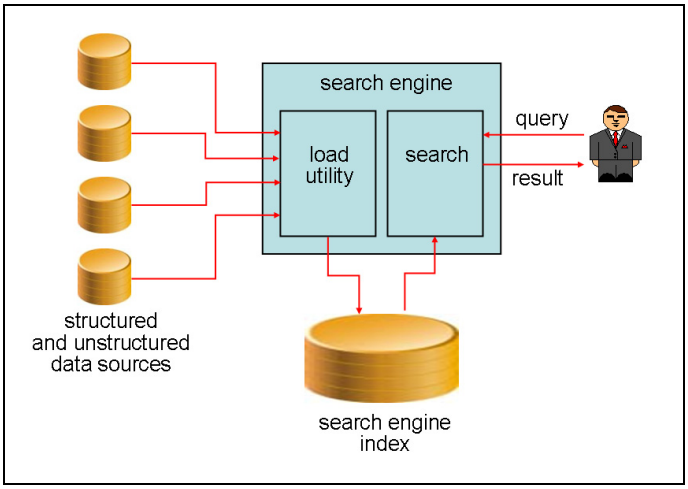

Figure 1 *A high-level architecture of a search engine*

Copyright © 2011 R20/Consultancy, All Rights Reserved.

The first key module of a search engine is the load utility for loading data from various data sources into the data store of the search engine. Load utilities are normally capable of high-speed loading, because in a search environment large amounts of data must be loaded regularly.

A search engine's data store is different from a traditional database. Where the latter consists of data distributed over multiple tables plus indexes to improve query performance, the data store of a search engine primarily consists of an advanced index. Indexes in databases contain pointers to the records in a table. The data store of a search engine is one large index where keywords point to the documents in which they appear.

The index of a search engine contains more data than the index of a traditional database. The index contains enough data so that it can execute queries without having to access the original data sources. In other words, a search engine can operate independently, and the processing of its queries won't interfere with the processing on the original data sources.

The second key module of a search engine is a search module that is able to interpret the user queries and to find the relevant documents in the database. Queries are keyword-oriented. This means users can look for those documents in which particular keywords appear. The result of such a query is normally a set of hits, where each hit is a reference to a document.

Two types of search engines can be identified: *web search engines* and *enterprise search engine*. The former category of search engines is designed for allowing users to find documents on the Web. Well-known examples are Google, Yahoo!, Bing, and CloudView. Enterprise search engines are used within organizations for allowing their own users to search for documents and data.

Comparison of Search Engines and Database Servers – In a way, the architecture of a search engine as presented in Figure 1 resembles the architecture of a classic database server. Both have features for importing or inserting data in their respective data stores, and both offer query capabilities. But that's probably where the comparison ends, because numerous differences exist. A few of those differences are described here:

Keyword-oriented: Most of the data stored in classic database servers is organized in tables and records. The data stored by a search engine is normally organized around keywords. All these keywords appear in one large index and they point to all the documents in which they appear. For example, a particular customer name might point to all the invoices of that customer, all his email messages, the list of returned products, or all the data on a webpage.

Flexible Data Model: In a database server, different types of data elements are stored in different tables, for example, customer data is stored in the Customer table, and product data in the Product table. All the data in a search engine is stored in one sophisticated index; using SQL terminology, it's almost as if all the data is stored in one large table. The consequence is that each query is aimed at that one index, thus at all the documents.

Before data can be stored in a classical database, the tables and columns must be designed. A database is not like one big container in which any type of data can be dumped. So, when a new type of data needs to be stored, table structures must be designed and implemented. This

Copyright © 2011 R20/Consultancy, All Rights Reserved.

is not required for a search engine. All the documents, regardless of what they represent, invoices, patient files, or call detail records, can be stored in the index of a search engine without having to define tables. This means that if documents of a new type must be added, they can be loaded straightaway, no design or development effort is required to make this possible. In short, the index of a search engine doesn't work with a conventional data model and is therefore more flexible.

Search: Database servers are excellent for processing queries on structured data. Search engines are developed for querying unstructured data. Their queries are always keyword-based. The result of a search-based query is a set of (pointers to) documents in which the keyword appears. An example of a query is: "Get all the documents that contain the keywords 'sales' and 'Boston'." In addition, the results show where those keywords appear in the documents. But it must be said that more and more database servers offer features for storing unstructured data and for search-based queries. They can process queries such as "Find those texts in which the word Boston appears." Some can even do queries such as "Find those pieces of text in which two words appear close together." In other words, the difference between the search capabilities of database servers and search engines is becoming smaller.

Semantically Enriched Search: Most search technologies are good at locating words in documents. It's quite valuable if a result is returned showing that a word appears in a number of documents, but will that result include documents in which synonyms of the search term are used, or different spellings, misspellings, or translations? For example, if we look for the word 'bike,' will the search engine also find the documents including the terms bicycle, bikes, byke, bicyclette (French), transporter, and so on?

To be able to do this, a search engine must *semantically process* the documents. This is done during the loading process. If we take the example of the bike again, to be able to find that larger set of documents, the load utility of the search engine has to know what the synonyms are of the word bike, how bike is spelled in other languages, what the conjugations of bike are, and so on. By semantically processing the data, more meaning is added, i.e. by semantically processing the data a search engine 'understands' the keywords better. The effect is that searching the data will lead to better results and that means the chance increases that users will find what they are looking for.

We call this search capability, in which the results are created by semantically processing the documents, *semantically-enriched search*. The purpose of this different name is to distinguish it from pure string-based search, which we will call *search*. Note that not all search engines support semantically- enriched search nor do most current database servers.

Deriving Structured Data from Unstructured Data: With search engines, structured data can be extracted from unstructured data when loading the data. For example, all the locations or customer names mentioned in a document, the language in which a piece of text is written, and the sentiment of a text can be derived (more on this topic in Section 6.2). These derived pieces of data are structured data. When derived, they can be used for reporting and analytics. In fact, the derived structured data can be combined with the original structured data. If semantically enriched searching is supported, the amount and the quality of the derivable structured data increases. Deriving structured data from unstructured data is clearly not a feature supported by database servers.

Copyright © 2011 R20/Consultancy, All Rights Reserved.

Dynamic Relationships: Database servers and search engines differ substantially in the way they handle relationships between (business) objects and the level of flexibility of those relationships. Imagine we want to find all the data we have on the customer called Mays Electronics; this is an example of 360° reporting. In a traditional database environment this implies that we must query all the records in all the tables in all the systems in which customer data is stored. In principle, this can only be done if key values have been assigned to customers and if they have been assigned consistently in every system. But if key values have not been assigned, the only other way to answer the query is by trying to locate those records based on customer name and/or customer address. But do we know for sure that the customer names are spelled correctly and consistently in every table of every system? What will happen if in some systems the name Mayes Electronics is used, or Mays Elecs, or just Mays? In that case, the chances are high that not everything we know about the customer will be found.

Another concern is when users want to extend their report to include data from a new system. This will require a lot of work. It could well be that keys must be added to this extra system purely to make joins possible. Additionally, a lot of manual work is required to assign the right key values to the right customers.

A search engine with semantical processing features will have fewer problems with both issues, because of the way stored data is organized. In a classic database the invoices of a customer are stored in one table, his complaints in another, and his returned products in a third. And in each table, key values are used to represent the customers. As indicated, a search engine's data is not organized around tables and key values, but around keywords. Its index will, for example, contain the customer names and each of those names will point to the documents (invoices, complaints, returned products) in which those customers appear. With the semantic processing, the chances are that every time data is loaded that is related to Mays Electronics, the search engine will know that it's the same customer even if the name is spelled somewhat differently, and it will add the new documents to the already indexed customer. No new relationships must be defined.

This means that in a search engine, adding data from a new system is merely a matter of loading it. If Mays Electronics does appear in the data of this new system, the 'link' will be created. In other words, the amount of work involved in adding new data is minimal compared to the amount work required when database servers are used.

Summarizing, the relationships between objects in database servers are static by nature. They can be added and changed, but it involves a considerable amount of design and development work. The relationships in search engines are completely dynamic. They are created the moment they come into existence.

Relevancy ranking: In a classic database all the rows in a result have the same relevancy. Results can be sorted on specific columns, but whatever order is selected, it probably doesn't correlate the relevancy of the rows to the users. Search engines are different. They try to present the results (a set of documents) in such a way that (probably) the most relevant document is presented first, the second most relevant one is presented next, and so on. To support this, a search engine supports the concept of *relevancy ranking*. Developers are able to indicate the characteristics that determine the relevancy of documents in a query result.

Copyright © 2011 R20/Consultancy, All Rights Reserved.

Summary – Although database servers are powerful engines for managing data, their focus is on managing structured data, on running queries, and on dealing with transactions. Search engines manage structured plus unstructured data and they excel at running search-based queries; see Figure 2.

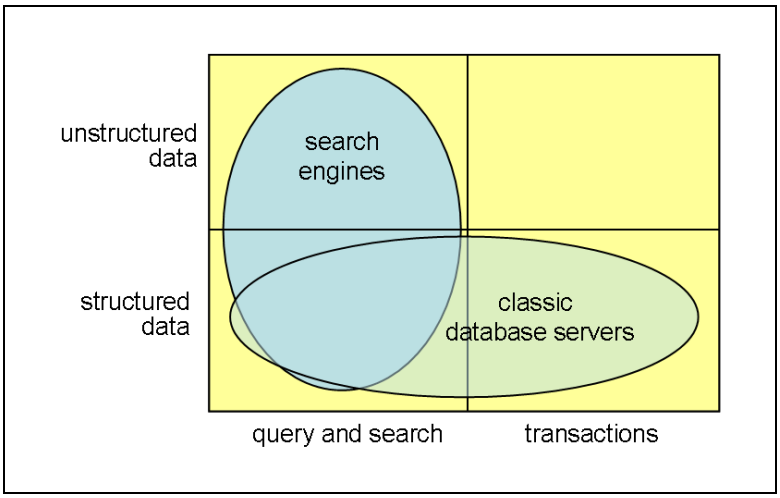

Figure 2 *Search engines versus classic database servers*

Section 2 introduces five new forms of reporting and analytics: *The New BI*. Table 1 lists which of the above characteristics of search engines are beneficial for the first four forms of reporting and analytics (the fifth one, long-tail reporting is addressed in Section 4). The more asterisks a table cell contains, the more beneficial the characteristic is. By looking at those asterisks, the table shows clearly that search engines, especially when they support semantically-enriched search, improve these new forms of reporting and analysis.

Characteristics of search engines	Unrestricted Ad-hoc Analysis	360° Reporting	Exploratory Analysis	Text-based analysis
Keyword-oriented	***	***	*****	*****
Flexible data model	*****	*****	*****	*****
Search	**	**	**	**
Semantically-enriched search	*****	*****	*****	*****
Deriving structured data from unstructured data	***	*****	***	*****
Dynamic relationships	*	*****	*****	*
Relevancy ranking	***	***	***	*
Query only	***	*	***	***

Table 1 *The importance of the characteristics of a search engine for The New BI*

Note: This section has looked at those differences between database servers and search engines from the perspective of how well the latter supports the new forms of business intelligence. This leads to an unbalanced view of database servers in general. If a comparison would have been made for other types of applications, database servers would have looked much stronger. The unbalanced comparison was intended because of the focus of this whitepaper, and should not be considered as an exhaustive comparison of the two technologies.

Copyright © 2011 R20/Consultancy, All Rights Reserved.

4 Mashup Technology in a Nutshell

The term *mashup*, which has become more and more of a house-hold term in the IT industry, is borrowed from the music industry, where it refers to a song created by blending two or more pre-recorded songs. In the IT industry the term mashup (or mashup application) refers to an application created by blending two or more existing applications.

In most situations mashups are new user interfaces developed on top of a number of existing applications. These applications can be internal, such as inventory and human resource applications; or external, such as Google Maps and Facebook; or applications running in the cloud, such as Salesforce and NetSuite. Most mashups are Web 2.0 based, meaning they exploit rich user-interface components. Although in most cases, mashups depend on existing applications they don't have to, they can be stand-alone applications.

Normally a distinction is made between *consumer* and *enterprise mashup applications*. The former group consists of applications developed for the general public, such as TaxiWiz, which figures out for a number of cities how much a cab ride is likely to cost by plotting a route using Google Maps; and Flickrvision, which shows the most recently uploaded photos from around the world in real-time on a map. Enterprise mashup applications are developed by and for organizations for internal use or for developing Internet-based applications.

Mashup technology can be used for various reasons:

- For developing a new and more efficient user interface on top of an existing application.
- For developing one integrated interface for two different applications to avoid forcing users to copy-and-paste data from one application to the other, bringing multi-source content together in a pertinent, contextual way (without conventional data or Web service integration).
- For developing low-cost, short-lived applications or applications for small groups of users, mashups offer a low-cost alternative for application development.
- For developing an alternative user-interface on top of an existing application for those users who use the application differently.

Most of the above applications fit the new business intelligence form named long-tail reporting like a glove. More on this in Section 7.

5 What is CloudView?

CloudView is a search engine developed and distributed by Exalead. This Paris-based company was founded in 2000 by François Bourdoncle and Patrice Bertin. Both have worked for Digital Equipment Corporation (DEC) in Palo Alto, California in the famed AltaVista project.

AltaVista is a web search engine. The AltaVista project was started in 1995 by DEC as a showcase project. The goal of the project was to demonstrate the power of DEC's new 64-bits platform. On December 15, 1995 the engine was released at altavista.digital.com. From day

Copyright © 2011 R20/Consultancy, All Rights Reserved.

one, AltaVista was a success. It had 300,000 hits on the very first day and more than 80 million hits per day two years later. The search engine changed hands a number of times in the last 15 years. Currently Yahoo! is the owner.

Bourdoncle and Bertin, who had worked at the AltaVista project, decided to start a new project to implement new ideas for developing a web search engine. They moved back to Paris where they had started their careers. Initially, it started as a research program of the École Nationale Supérieure des Mines de Paris. Later, Qualis, an investment company with offices in Paris and New York, made it possible to start the company Exalead. Together with students of the university in Paris, CloudView was developed.

The first commercial version of CloudView became available in 2003. On the outside, it was a web search engine comparable to any other. Like all search engines it allowed people to search for keywords on the web. This web search engine is still available; see http://www.exalead.com/search/. With respect to the number of documents indexed, CloudView ranks number four, after Google, Yahoo!, and Bing. On the inside, however, new technology was applied. And this is how CloudView distinguishes itself from the competition. More on this later in this whitepaper.

In 2004 the decision was made to release a separate product for enterprise search to help organizations search their own documents and data. The first version of this enterprise search version was released in 2006. The current version of CloudView is 5.1 and was released at the end of 2010.

On June 9, 2010 Exalead was bought by Dassault Systèmes (who specializes in product lifecycle management) from Qualis. They serve businesses of all sizes in all sectors, from automotive to fashion, from industrial equipment to consumer goods, pharmaceuticals, architecture, and services.

Currently, besides running the web search engine, Exalead offers the enterprise search engine CloudView (which is built on the same platform as the web search engine), and Mashup Builder, which together enable users to quickly develop intelligence applications based on search capabilities. Applications developed with Mashup Builder are created by mashing up existing data sources.

6 Characteristics of the CloudView Search Engine

Section 3 contains a high-level overview of the architecture of a generic search engine. The architecture of the CloudView search engine is in line with that description. This chapter describes the specific characteristics of CloudView's architecture and explains some of its unique features in detail.

Copyright © 2011 R20/Consultancy, All Rights Reserved.

6.1 The Structure of the CloudView Index

Keywords and Indexes – All the data stored by CloudView is stored as one large, sophisticated index. This index contains all the searchable keywords. For each keyword, it contains pointers to the original documents plus additional, descriptive data, such as synonyms, attributes, locations of the keyword in the documents (section, page, line), and statistical data, such as in the number of documents in which the keyword appears.

A CloudView index is fully self-supporting. This means that if users search on particular keywords, CloudView can respond by indicating in which documents those keywords appear, plus it can also show fragments of those documents–even if the original data source is offline. This means that a CloudView index is not the same as a classic index of a SQL database. The latter is just an index with pointers to records. Without the data in tables, a classic index has no value.

Data can be enriched in a CloudView index by using the semantic factory. Data can be imported from any source, including a PDF document, a Word document, an email, a web page, and an RSS feed. The load utility of CloudView locates all the keywords, processes them semantically, and extends the index.

Indexing Structured Data – One of the features that makes CloudView unique is that it's not limited to indexing unstructured data. It can also index data stored in classic databases. For example, all the payments of a customer stored as rows in a SQL database can be exported to become a document for CloudView, or all the data describing a product could become a document.

Conceptually, the way this works is simple. A SQL query is defined to retrieve relevant data from a database. The result is loaded into a CloudView index like any other document. Thus, instead of indexing individual records, documents are created in which data that logically belongs together, is stored together. For example, all the order records making up one order plus the order heading record might become one document. Or, all the invoices of one customer form one document. Any group of data elements that can be created with one SQL query can form a document.

This feature allows for mixing unstructured and structured data. The result is that when users search for the name of a customer, the result could contain, for example, data from emails, plus data from production databases, and from the central data warehouse. This is ideal for, for example, 360° reporting.

6.2 Semantic Processing of Documents

CloudView supports *semantically-enriched search*; see Section 3. Before keywords are added to the index, CloudView extracts as much meaning from the text as possible. For example, different spellings are determined, synonyms are found, and generalizations and specializations of the keywords are derived. This process is called *semantic processing*. The

Copyright © 2011 R20/Consultancy, All Rights Reserved.

more intelligent this process is, the more keywords are enriched and the better the answer to queries will be.

Semantic processing has been implemented in CloudView by allowing developers to specify multiple *semantic processing operations*, or, for short, semantic operations, that are applied when documents are loaded. CloudView supports a wide range of semantic operations. For each type of incoming document, developers can define which semantic operations should be applied. For this, CloudView supports a flow-like language. Each semantic operation is like a step in a flow. A flow specifies a serial stream of semantic operations. Such a stream is called a *pipeline* and is executed by the module called the *semantic factory*; see Figure 3. A semantic operation can be the call of a simple string manipulation function, for example, to remove the brackets and dashes in a telephone number, but it can also be a highly advanced function for sentiment analysis. So, while a document passes through the semantic factory, it is enriched with more meaning.

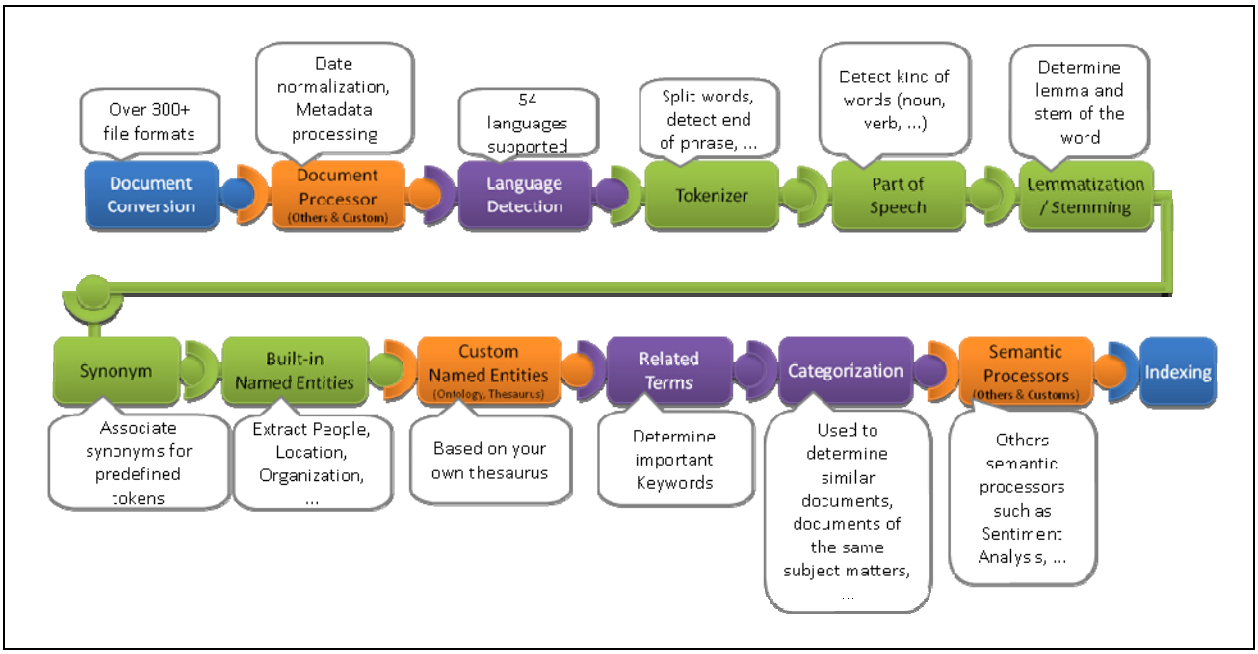

Figure 3 *The pipeline specifies the semantic operations executed by the semantic factory; source Exalead*

With semantic operations, structured values can be derived from the documents, such as the language in which it's written, or the tone of the document (positive or negative). Afterwards, documents can be searched based on those derived values.

CloudView supports a large set of powerful semantic operations. To give a feeling of its capabilities, some of these operations are described below. For a more extensive list, we refer to the manuals.

Language Detection – One of the supported semantic operations is language detection. Based on the words used in a document, CloudView can distinguish between more than fifty languages, including English, French, German, Spanish, Arabic, Chinese, and Dutch. The language detected is added to the document as an attribute and thus becomes available to the

Copyright © 2011 R20/Consultancy, All Rights Reserved.

other semantic operations. In the example of the hotel, where customers can leave special requests, it could be that everyone enters their requests in their own native language. In this case language detection is a vital semantic operation.

Synonym Association – The platform supports the use of standard and custom synonym lists (thesauri). The effect of synonym association is that looking for a keyword might lead to finding all the documents in which a keyword appears plus all the documents that contain a synonym of the keyword.

Location and People Extraction – Some pieces of text contain names that might be references to people, companies, or locations. This type of semantic operation, called 'named entity extraction,' is able to locate these names in the text. When found, they are added to the document as structured attributes. For example, a complaint might contain the city name of the store where the customer was mistreated. Or, a tweet might contain the name of a department or employee.

In addition, CloudView supports features that can distinguish objects with the same name from each other. For example, by 'reading' the rest of a sentence it can determine whether the word 'shell' refers to an oil company or to something we find on the beach. This is done by studying the context.

After those names are found, it's easy for CloudView to answer questions, such as "Show me all the documents in which Michael Phelps appears plus the location Manchester", or "Find all the complaints related to our store in Boston." Combined with all the other semantic operations, this is an important feature for bringing together data related to, for example, the same customer, even if the customer names are spelled differently or incorrectly.

Lemmatization and Stemming – Words in texts appear in different forms. In many languages many conjugations can exist of a specific verb. For example, conjugations of the verb 'to walk' include walk and walked, and of the Dutch verb wandelen (which means walking) the conjugations wandel, wandel, and wandelden exist. And in many languages singular and plural versions exists for nouns. By applying lemmatization and stemming, it becomes possible for CloudView to find any form of a keyword when searched for.

Categorization – Documents can be classified in two basic ways. One way is to use specific rules or ontology's to classify and categorize data. For example, a simple rule could categorize documents based on their length: short, medium, or long; invoices extracted from databases might be categorized based on the total amount of the invoice; and received emails can be categorized based on select content: complaint, question, or other. Another way is to dynamically categorize documents based on discovered attributes, like extracted entities or related terms, or calculated attributes, like sums or averages. These categories and subcategories, and the data clusters built upon them, are called 'facets.' These facets are structured data and can be used for reporting and analysis.

Sentiment Analysis – With this type of semantic operation CloudView tries to determine and extract the subjective information in the text. What is the attitude of the writer or what the tone of the text. Imagine that an organization analyses published tweets that concern themselves. The first thing the organization wants to know is whether it's a positive or negative tweet. For

Copyright © 2011 R20/Consultancy, All Rights Reserved.

example, the tweet "I will never stay at this hotel again" should evidently be classified as negative, whereas the tweet "I will definitely be back" should be considered positive. Sentiment analysis is not easy, especially because some writers can be cynical or sarcastic. So, the result is not always 100% perfect.

String Manipulations – Besides all the above semantic operations, CloudView supports all kinds of string manipulation functions. Strings can be split, concatenated, and so on.

A pipeline of semantic operations will always contain multiple operations. Work on such a pipeline will never end, because it can always be improved, plus use of words and terms changes over time.

6.3 Search and Query Features

Most data warehouses and data marts are created with SQL or MDX databases. This means that to query the data in those databases, SQL or MDX must be used. CloudView has its own query language, a search language supporting semantically-enriched searches. The search features of CloudView are very extensive. In this respect, this product can compete with any other search engine. For an in-depth understanding of this language we refer to the CloudView manuals.

The result of a search is not just a straightforward set of rows, but a set of documents ranked for their potential relevancy to the user and organized by facets (clusters) built upon implicit and explicit attributes, an indication of the total number of documents in the result, set plus for each hit, attribute values. These attributes are especially important for the business intelligence world. Such an attribute value could be the derived language in which the document is written, another could be the tone of the document, and a third an indication of whether our company name appears in the document.

Documents can also be searched based on those attributes. For instance, users can click on facets (presented in textual or visual form) to retrieve all the documents written in German and that are positive in tone (a process called faceted navigation or parametric search). Mashup Builder, described in Section 7, is capable of using there facets to create business graphs.

6.4 Accessing CloudView

CloudView supports several technical interfaces for accessing data. One is a out-of-the-box HTML UI for enterprise search. It is fully customizable and offers basic and advanced search features. It's comparable to the well-known Google interface: one input field: a search text box.

CloudView also offers programming interfaces (APIs) for Java and .NET. These interfaces can be used by applications and tools to extract data from the CloudView index. The third option is the Mashup Builder, the drag-and-drop tool for constructing business applications, including BI applications. It's discussed in the next chapter.

Copyright © 2011 R20/Consultancy, All Rights Reserved.

6.5 Scalability of the CloudView Engine

As indicated in Section 5, CloudView was initially designed to be a web search engine scalable enough to handle all the webpages of the World Wide Web. Currently, that engine has indexed over sixteen billion documents. To be able to do this, its architecture is highly efficient and scalable. CloudView for enterprise search has a comparable architecture, and, therefore, has inherited the scalability and functionality of the web search engine.

One of the features that makes CloudView scale is that a CloudView index can be sliced. This means that the index is broken into *slices* and those slices are distributed over multiple disks and hosts. As a result, multiple systems can access the index slices in parallel, improving the response times of queries. This is comparable to partitioning of tables in SQL database servers. The names of the keywords are used for determining which part of the index ends up in which slice.

Slices can also be *replicated*. One or more replicas of a slice are stored on separate disks. This feature has three advantages. First, it increases the availability of the data, because if one replica is not available, one of the others can be queried instead. Secondly, it also allows for running even more queries in parallel: queries accessing the same data are sent to different slices. And thirdly, it allows for fast loading of new data, because different slices can be loaded in parallel.

It is a framework for parallel processing in a distributed environment that shares characteristics with the *Bigtable/MapReduce* approach[1].

6.6 Importing and Refreshing Data

As data is added to or changed in the data sources, the CloudView index has to be refreshed. For this, CloudView supports technology that keeps an eye on whether new data has been inserted or altered. It primarily uses a push model. To each data source a process is attached that analyzes the original data source and looks for new data. These processes are called *connectors* in CloudView. Depending on the source itself, the connectors poll the data source for new data or the new data is pushed by the source to the connector. When a connector has received new data, it transforms it in a document, and places it into a *task queue* on the Push server.

Separate modules, called *analyzers*, pick documents from the task queue and run the appropriate pipelines for them. Multiple analyzers can run pipelines in parallel. The resulting processed documents, which represent changes to be made to the index, are received by the *Index Builders* attached to particular slices. It is the job of each Index Builder to aggregate

[1] J. Dean and S. Ghemawat, *'MapReduce: Simplified Data Processing on Large Clusters'*, in Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation - Volume 6, San Francisco, CA, December 06 - 08, 2004.

Copyright © 2011 R20/Consultancy, All Rights Reserved.

updates for its slice from all the analyzers, and to load those updates to the slice. As indicated, loading of new data on different slices is executed in parallel.

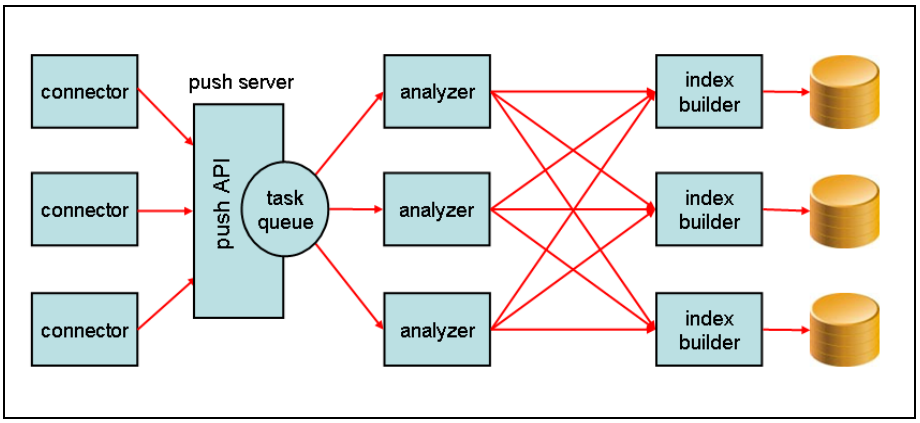

Figure 4 *The highly parallelized internal architecture of CloudView*

As indicated as well, queries are also run in parallel. What's more, querying and indexing are fully concurrent processes, eliminating the need to lock parts of the index during query processing. In comparison to classic database servers, it is a very non-disruptive means of performing sequential updates. If a particular update has not been fully committed to a particular slice, queries are still processed as normal: the specific update in question may simply not be reflected in the full-index result set returned to the user. However, the gains in performance and availability enabled by this architecture make concurrent querying and updating across index slices and replicas an excellent compromise for the New BI.

To summarize, CloudView has a highly parallelized, multi-threaded architecture, allowing for loading massive amounts of new data, and offering a high level of availability for queries.

7 Mashup Builder

Exalead's second technology discussed in this whitepaper is *Mashup Builder*. Mashup Builder is a high-level *mashup tool* for quickly developing applications that need to integrate data stored in CloudView, websites, SQL databases, RSS feeds, and other sources; see Figure 5.

Mashup Builder offers two special features. First, the tool can fully exploit the features of CloudView so that it becomes easy to add search capabilities to the mashup applications. This class of applications, where search technology is the driving component, is called *search-based applications* (SBA). Imagine an application with which users can look up customer information. In a more traditional application the user sees a number of input fields for entering data, such as name and address. The application then tries to find the right customers. In most cases, nothing will be found if the entered values are not correct. One misspelling in a word could lead to an empty result. With a search-based application, the user can enter anything he knows about the customer, and the search engine will try to find those customers that resemble the keywords entered by the user. It's a more free format search. This would be a typical search-based application.

Copyright © 2011 R20/Consultancy, All Rights Reserved.

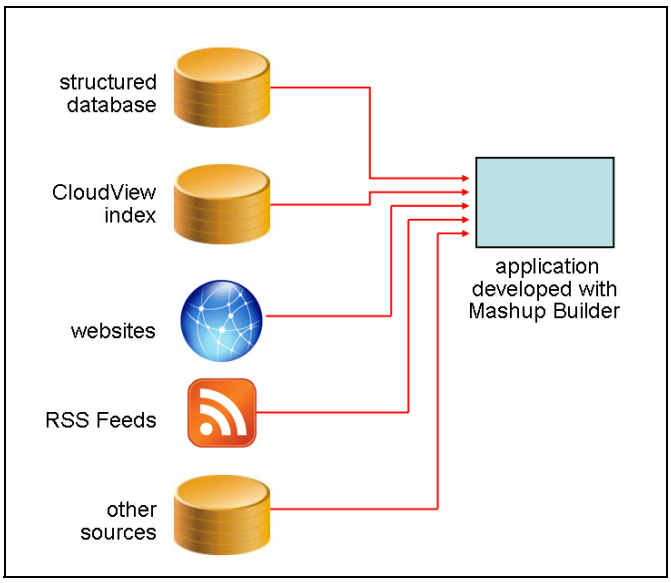

Figure 5 *With Mashup Builder applications can be developed that integrate data from a wide range of internal and external sources*

The second special feature of Mashup Builder is that it can present data as business graphs, such as pie charts and bar charts. In this respect it resembles a reporting or dashboarding tool and makes it suitable to be deployed in business intelligence systems. One can decide which facets one wishes to present visually. Again, facets can be generated using dynamic clustering or they can be specified in a custom ontology. In addition, in CloudView hierarchical relationships between keywords can be defined. For example, one can define that Boston is a city belonging to the state of Massachusetts and Massachusetts belongs to the USA. If this is repeated for most cities, Mashup Builder will be able to exploit those specifications, and will make it possible for end users to drill down and roll up data. For example, if sales data per state is presented, the end user can click on one of those states in the business graph to see the more detailed sales data per city within that state.

Figure 6 *Example of a business intelligence application developed with Mashup Builder*

Copyright © 2011 R20/Consultancy, All Rights Reserved.

Although the tool is high-level and doesn't require a lot of training hours to master, it's not a self-service tool for end-users. It's rather a drag-and-drop tool for professional developers to create self-service applications for end users, including long-tail applications for small numbers of users and/or short-term usage.

To summarize, by combining these two special features, developers can quickly develop search-based BI applications with Mashup Builder, in which free format search capabilities can be combined with traditional drill-down and roll-up capabilities normally found in business intelligence reporting and analysis tools, and where structured and unstructured data sources and external and internal data sources can be queried.

8 BI Application Areas of CloudView and Mashup Builder

This chapter describes how CloudView and Mashup Builder can be exploited to support the new forms of reporting and analytics as described in Chapter 2:

- Unrestricted ad-hoc analysis
- 360° reporting
- Exploratory analysis
- Text-based analysis
- Long-tail reporting

8.1 Unrestricted Ad-Hoc Analysis

In an unrestricted ad-hoc analysis environment, users need access to:

- Operational data from the operational sources
- Aggregated data from the data warehouse environment
- Unstructured data
- Data from external sources

The first one is very important, because this form of analysis is primarily used by operational management, so they need access to the latest state of the data. CloudView's semantic factory is able to merge data from all these sources; see Figure 7.

The semantically rich search capabilities of CloudView will allow management to use the search-based interface to browse, navigate, and query all that data in an unrestricted fashion. There is no need to predefine tables and relationships.

Copyright © 2011 R20/Consultancy, All Rights Reserved.

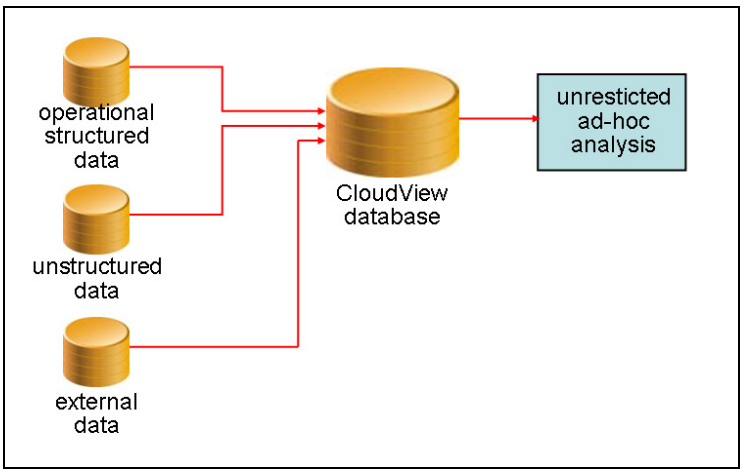

Figure 7 *By loading data from operational systems, unstructured data sources and external sources CloudView becomes a first-rate instrument for unrestricted ad-hoc analysis*

8.2 360° Reporting

360° Reporting means being able to paint a full picture of a business object, such as a customer, bank account, or flight. To be able to do this, users need access to at least the following data sources (see also Figure 8):

- Aggregated data from the data warehouse environment
- Historical data from the data warehouse environment
- Operational data from the operational sources
- Unstructured data
- Data from external sources

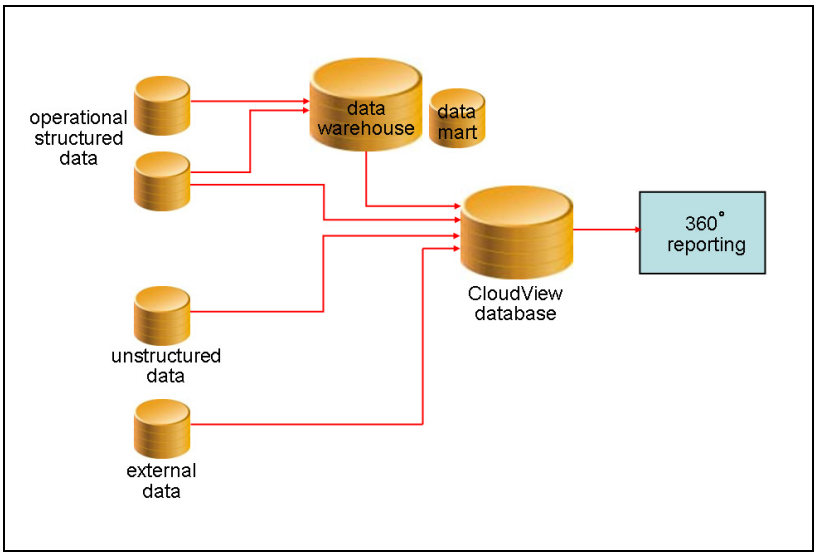

Figure 8 *By loading data from the data warehouse, unstructured data sources and external sources, CloudView can support 360° reporting*

When loading from a wide range of sources, data is integrated based on semantic relationships. For example, a search for a specific customer could return straightforward

Copyright © 2011 R20/Consultancy, All Rights Reserved.

address data coming from the data warehouse, his order history also coming from the data warehouse, an indication of the number of positive and negative complaints emailed by them, relationships with other customers, and so on. In other words, it's almost as if the semantic factory automatically creates a 360° view of a business object in the index.

Afterwards, it's easy to add more data sources. This won't require any design or development work. The only thing to be done is that the new data source must be loaded and has to be processed by the semantic factory.

8.3 Exploratory Analysis

In a SQL database, data is stored in tables. Data from different tables can be integrated if the correct columns have been added to those tables. In other words, relationships between tables must be predefined. From time to time users need to freely browse and analyze their data. They need to relate data elements for which no relationship has been predefined. In other words, they want to exploit unknown relationships. This is exploratory analysis.

In a CloudView index, data is stored in a keyword-centric style. Those keywords point to all the documents in which they appear. When new documents are loaded, the loading utility analyzes them and determines which keywords are new and which existing keywords should be extended with pointers to those new documents. This means that a search for a keyword returns all the structured and unstructured data related to that keyword. For example, a search for the word Boston, might lead to a result that consists of the invoices sent to customers based in Boston (structured data), the deliveries to shops based in Boston (structured data), but also all the emails sent out by sales people in which the word Boston appears (unstructured data), and all the contracts with suppliers that contain the word Boston (unstructured data). Important to understand is that no need exists to define relationships.

We will explain this by using some simple examples. A SQL query is normally aimed at a specific set of columns, for example:

```
SELECT   *
FROM     CUSTOMERS
WHERE    CITY = 'Boston'
```

The result of this query returns all the customers based in Boston, and the only data we get is the data stored in the Customers table.

If we would use SQL terminology, with the CloudView query language all the columns in every table can be accessed at the same time. In CloudView it's almost as if we ask:

```
SELECT   *
FROM     ALL_TABLES
WHERE    * = 'Boston'
```

Copyright © 2011 R20/Consultancy, All Rights Reserved.

The specification `ALL_TABLES` means: Execute the query in all tables in the database; and the condition `* = 'Boston'` means: Find all the records where the value of any column equals Boston.

Accessing all the tables in a SQL database with one query is just not possible, but looking for the value Boston in all the columns in a particular table is. In fact, we have two options. We could do it like this:

```
SELECT   *
FROM     CUSTOMERS
WHERE    CITY = 'Boston' OR COL1 = 'Boston' = OR COL2 = 'Boston' OR … OR COLN = 'Boston'
```

or like this:

```
SELECT   *
FROM     CUSTOMERS
WHERE    CONTAINS('Boston', CITY || COL1 || COL2 || … || COLN)
```

Both examples require that the developer knows in which column the value Boston might appear, because all the columns must be explicitly specified in both queries. The developer has to be exhaustive. In addition, both queries will be hard to optimize for most SQL database servers. In fact, it will probably lead to a scan of the whole Customers table.

The CloudView query, on the other hand, returns all the rows from all the tables in which the value Boston appears. Developers don't have to know in which column the value might appear. This flexible feature makes CloudView suitable for exploratory analysis. It allows users to freely navigate the data.

Additionally, CloudView will also search for alternative names of Boston, such as The Puritan City or Bean Town. If there are documents in which those nick names have been used instead of the original name, they will also be found. This would be hard to do with a SQL-based environment. In fact, all the search features make CloudView ideal for exploratory analysis. It will find documents where more specific terms are used, such as a neighborhood of Boston, for example Beacon Hill or Charlestown, or more generic terms, such as Massachusetts.

To summarize, the ability to search for keywords in any data source (once it is loaded into CloudView) makes CloudView a tool for exploratory analysis.

8.4 Text-based Analysis

Regardless of how powerful reporting and analytical tools are, the results they can show are limited by the accessible data. The contents of most data warehouses are restricted to structured data coming from operational databases. Because no unstructured data is available in most data warehouses, it can't be analyzed, therefore limiting the analytical and reporting capabilities of the business.

Copyright © 2011 R20/Consultancy, All Rights Reserved.

Text-based analysis is needed when organizations require:

- Reporting on unstructured data
- Statistical modeling on unstructured data
- Dashboarding on unstructured data
- Combining structured data with unstructured data

In addition, when they prefer to do all this using their current reporting and analytical tools, structured data must be derived from unstructured data.

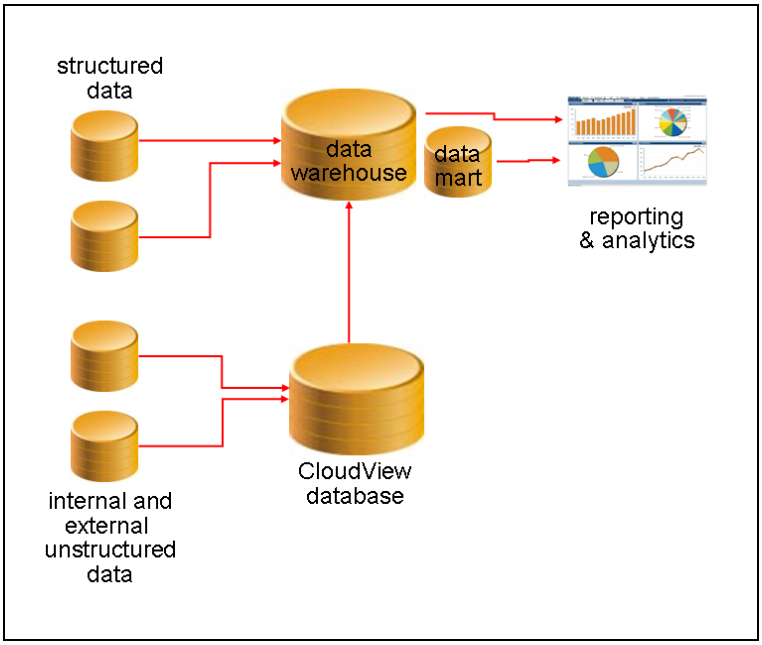

Figure 9 *Using CloudView, structured data can be derived from unstructured data sources and loaded into the data warehouse to make it available for reporting and analytics*

In CloudView, extracting structured data is done using the semantic factory; see Section 6.2. For example, by using sentiment analysis, pieces of text can be evaluated. The result might be an indication of the tone of the text, is it positive or negative? Another example is that product names can be extracted from text. In both cases, the result has the form of structured data and can be used for analysis and reporting.

The structured data derived by CloudView from unstructured data, can be exported and loaded in a data warehouse; see Figure 9 (note, however, that the data may require some further transformation to ensure alignment with the data warehouse's schema). After storing it in a data warehouse, all the reporting and analytical tools can be used to analyze the unstructured data in the same way the structured data is analyzed.

To summarize, CloudView makes it possible to extract structured data from unstructured data and the former can be stored in a data warehouse, where it is integrated with the existing structured data, and where it becomes available for reporting and analytics.

Copyright © 2011 R20/Consultancy, All Rights Reserved.

8.5 Long-Tail Reporting

Mashup Builder can be used for quickly developing long-tail reporting applications. The product allows for integrating data from many different sources, including external websites and RSS feeds. It can fully exploit the semantically rich search features of CloudView, and it allows for the development of Web 2.0-based, user-friendly business intelligence applications with typical business intelligences features such as drill-down and roll-up.

9 Advantages of CloudView

To summarize, together Exalead's CloudView and Mashup Builder offer the following advantages:

- CloudView supports a highly scalable architecture for handling large amounts of data and users concurrently.
- CloudView offers a powerful set of semantic operations for enriching the search capabilities and for deriving structured data from unstructured data.
- Business intelligence systems can be extended with CloudView for implementing *The New BI*: unrestricted ad-hoc reporting, 360° reporting, exploratory analysis, and text-based analysis.
- Mashup Builder is a mashup tool aimed at developing search-based applications with reporting capabilities, i.e. it's designed to support long-tail reporting.

10 Case Studies

This section contains two brief case studies of organizations using CloudView and Mashup Builder.

GEFCO – GEFCO is a key player in the market of integrated logistics for manufacturers. The company provides multimodal transport and end-to-end supply chain services for industrial clients in the automotive, two-wheel vehicle, electronics, retail, and personal care sectors. A wholly-owned subsidiary of PSA Peugeot Citroën, GEFCO is one of Europe's top 10 logistics groups, generating revenue of €3.5 billion in 2008. GEFCO's 10,000 employees serve customers in more than 100 countries on 6 continents.

A few years ago, GEFCO's vehicle track and trace service for automotive manufacturers began to falter under a heavy load. They experienced severe performance and usability issues with their previous database track and trace portal. Complex queries took minutes or even hours to process and data latency was approximately 24 hours. The system had to be closed every day for a couple of hours because of internal transaction processing.

Copyright © 2011 R20/Consultancy, All Rights Reserved.

GEFCO switched to CloudView for a makeover that boosted performance, enhanced usability, and enabled operational business intelligence—at half the cost of the legacy solution.

Some facts: GEFCO carries 3.5 million vehicles every year. This leads to 100,000 daily transactions in automobile transport on 600,000 vehicles. The current system developed with CloudView has 4,000 potential internal and external users. The size of the transactional database is 3 Terabytes. The CloudView index is refreshed every 15 minutes, and is therefore close to real time.

The benefits of CloudView to GEFCO are:

- Usability: Simple and intuitive search capabilities that require no training. A single text box, faceted navigation, and visual reporting enable customers and service staff to track and trace the daily movements of 600,000 vehicles with the speed and simplicity of the Web.
- Scalability and performance: The CloudView-based solution supports several thousands of users concurrently, indexes data for 100,000 daily transactions on car transportation, queries with sub-second response times against 3 Terabytes of (near) real-time data.
- Access to operational data for decision-making: The CloudView index is updated every 15 minutes. This allows users to search in near real-time data with no impact on the operational databases.

Akerys – Toulouse-based Akerys is a leader in rental investments and commercialization of new housing. Akerys offers four lines of service: property development, commercialization of new housing, brokerage (insurance, credit and financial products), and real estate services. They have over 100,000 customers, and more than 2,000 employees. In June 2009, their revenue was almost €600 million.

Their business challenge was four-fold:

- There was no unified view of clients. This was due to several mergers and to the fact that their four lines of business were completely differentiated from each other.
- Their was no unified view of their market. The Akerys' employees needed a more precise and up-to-date view of their business market. This would help them better define prices and improve decision-making.
- Clients needed unified information access. Akerys had to support thousands of potential buyers looking up specific pieces of data before deciding to invest. By streamlining access to key data it would become easier for users to intuitively locate the important internal and external data they needed to guide their investment decisions.
- Real estate developers (including Akerys itself) needed some kind of 'real estate barometer' business intelligence tool.

Akerys decided to use CloudView as the core of the solution. CloudView is currently used to aggregate internal data for a complete customer view, and to bring together data from their own production systems and from external websites to provide a comprehensive view and near-real-time analysis of their business market.

Copyright © 2011 R20/Consultancy, All Rights Reserved.

For market analysis, the required data is processed through three main stages. First, the web is crawled to capture raw data. Then, via analysis and optimization, the data is filtered, structured, and aligned with internal data. Finally, this aggregated data is presented using dynamic business charts, where it can be filtered based on geographic region, type of listing, and other characteristics. Statistics on a population (evaluation of price/m^2, evolution in prices or listing volume over specified time frames or locations, etc.) can be extracted as well.

CloudView is also used for data archiving to better track changes and detect trends in the market. Because data is brought together from multiple (internal and external) sources, users can access quantitative data, a photo library, and data management tools all in one place. Figure 10 shows examples of their applications.

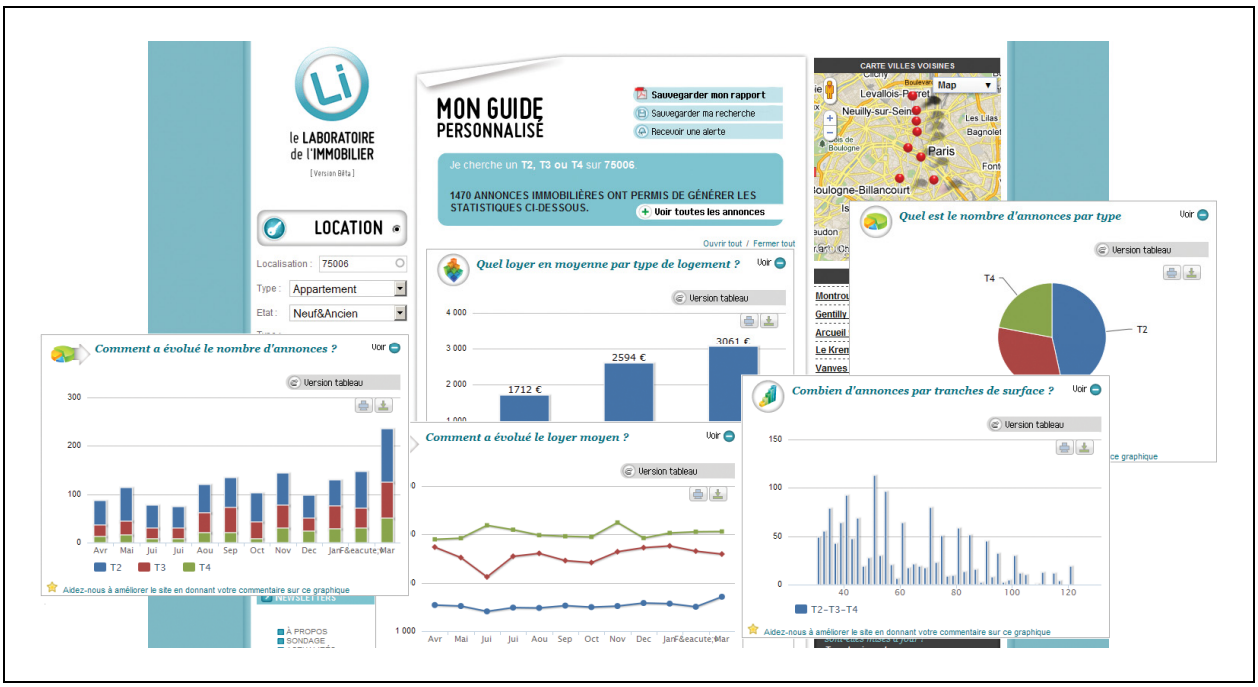

Figure 10 *Examples of the Akerys business intelligence applications*

Copyright © 2011 R20/Consultancy, All Rights Reserved.

About the Author Rick F. van der Lans

Rick F. van der Lans is an independent analyst, consultant, author and lecturer specializing in data warehousing, business intelligence, service oriented architectures, and database technology. He works for R20/Consultancy (www.r20.nl), a consultancy company he founded in 1987.

Rick is chairman of the annual European Data Warehouse and Business Intelligence Conference (organized in London), chairman of the BI event[2] in The Netherlands, and he writes for the B-eye-Network[3]. He introduced the business intelligence architecture called the *Data Delivery Platform* in 2009 in a number of articles[4] all published on BeyeNetwork.com.

He has written several books on SQL. His popular *Introduction to SQL*[5] was the first English book on the market in 1987 devoted entirely to SQL. After more than twenty years, this book is still being sold, and has been translated in several languages, including Chinese, German, and Italian.

For more information please visit www.r20.nl, or email to rick@r20.nl. You can also get in touch with him via LinkedIn (http://www.linkedin.com/pub/rick-van-der-lans/9/207/223) and via Twitter (http://twitter.com/Rick_vanderlans.

About Exalead S.A.

Founded in 2000 by Search engine pioneers, Exalead is the leading search-based application platform provider to business and government. Exalead's worldwide client base includes leading companies such as PricewaterhouseCooper, ViaMichelin, GEFCO, WorldBank, and Sanofi Pasteur, and more than 100 million unique users a month use Exalead's technology for search. Today, Exalead is reshaping the digital content landscape with its platform, Exalead CloudView, which uses advanced semantic technologies to bring structure, meaning and accessibility to previously unused or under-used data in the new hybrid enterprise and Web information cloud. CloudView collects data from virtually any source, in any format, and transforms it into structured, pervasive, contextualized building blocks of business information that can be directly searched and queried, or used as the foundation for a new breed of lean, innovative information access applications.

Exalead was acquired by Dassault Systèmes in June 2010. Exalead has offices in Paris, San Francisco, Glasgow, London, Amsterdam, Milan, and Frankfurt.

---

[2] See http://www.bi-event.nl/59857
[3] See http://www.b-eye-network.com/channels/5087/articles/
[4] See http://www.b-eye-network.com/channels/5087/view/12495
[5] See http://www.amazon.com/Introduction-SQL-Mastering-Relational-Database/dp/0321305965/ref=sr_1_1?ie=UTF8&s=books&qid=1268730173&sr=8-1

Copyright © 2011 R20/Consultancy, All Rights Reserved.