

Investigative Analytics using Xurmo's Hadoop-Based Platform

Combining the Best of NoSQL, SQL, and SQL-on-Hadoop

A Technical Whitepaper



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

February 2015



Sponsored by



XURMO
Configure Intelligence



Copyright © 2015 R20/Consultancy. All rights reserved. Xurmo is a trademark of Xurmo Technologies Pvt. Ltd.. Trademarks of companies referenced in this document are the sole property of their respective owners.



Contents

01	Management Summary	01
02	The Need for Investigative Analytics	03
03	Requirements for Investigative Analytics	06
04	The Ever-Growing Mountain of Data	10
05	Overview of Current Data Store Technologies	14
5.1	SQL as Data Store technology for Investigative Analytics	15
5.2	NoSQL as Data Store technology for Investigative Analytics	19
5.3	SQL-on-Hadoop as Data Store technology for Investigative Analytics	22
06	Xurmo Explained	24
6.1	Xurmo's Unique Concepts	25
6.2	Xurmo's Query Capabilities	27
6.3	Xurmo's Internal Architecture	35
07	About the Author Rick F. van der Lans	38
08	About Xurmo Technologies	39



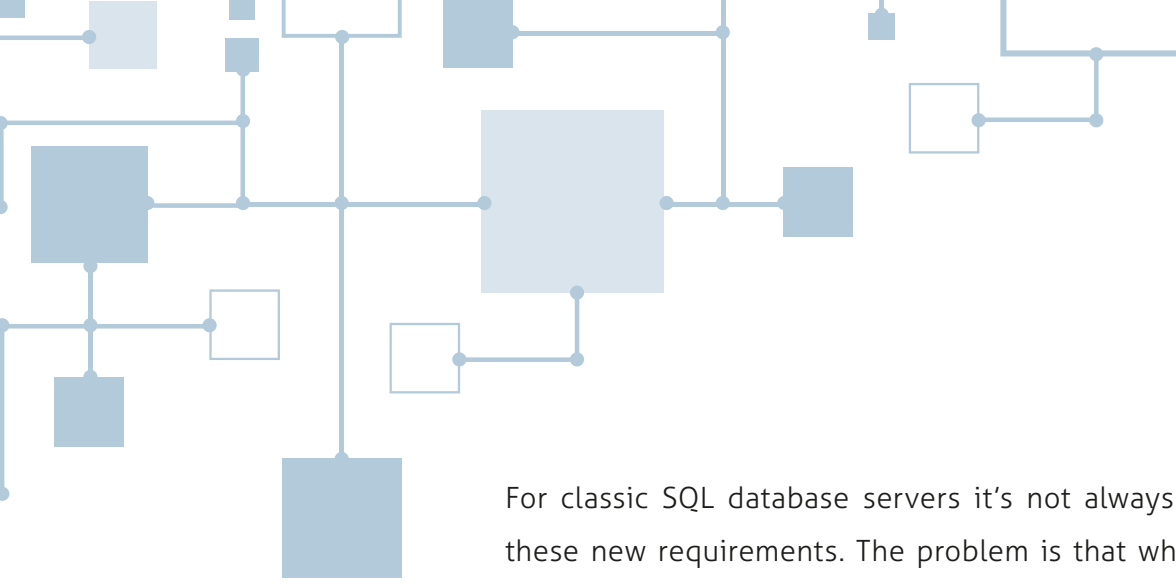
01 | Management Summary

This whitepaper describes the unique architecture and features of Xurmo. This product is designed to support investigative analytics (data discovery) on big data. Investigative analytics is an activity where users can freely navigate the data, even over un-defined relations. Xurmo uses a highly unique storage model, and its internal architecture is developed on Hadoop's modern-day distributed storage technology and Spark's highly scalable processing technology.

The analytical and reporting demands of managers and decision makers continue to change. A new form of analytics is called *investigative analytics* (or *data discovery*). Investigative analytics allows users to freely navigate and query the data without any restrictions imposed by database structures. The goal of such an exercise is to find a solution to a badly defined problem, such as “What could be the reason that sales in Boston are down?” The reason could be anything, ranging from late deliveries to employee-related problems. Users must not be limited in their investigative capabilities by the data structures of the tables or the pre-defined relationships between tables. Real investigative analytics makes it easy to analyze data across predefined as well as non-defined relationships. It allows new data sources to be integrated instantly.

Investigative analytics allows users to freely navigate and query the data without any restrictions imposed by database structures.

The requirements for tools for investigative analytics are: free form querying, access to any data source, instant data integration, advanced analytical features, high performance, and access to detailed data.



For classic SQL database servers it's not always easy to meet these new requirements. The problem is that when users work with SQL, they must know the data structures of the tables and columns and they must understand the undefined relationships between tables. An extra problem is that the queries are not adapted automatically when the data sources change. The same applies for the newer data storage technologies such as *NoSQL* and *SQL-on-Hadoop*. Although these technologies store data differently and use different query languages, the problems meeting the requirements for investigative analytics are similar.

This whitepaper describes *Xurmo*, a tool designed specifically for investigative analytics. It can be classified as a SQL-based product. However, internal storage of data is totally different from the familiar storage techniques. *Xurmo* helps users to formulate their queries in whatever direction they are heading. It supports advanced analytical capabilities, such as sampling, classification, regression, and clustering, and allows for the definition and embedding of *analytical workflows*. The product comes with its own database technology, which is based on Hadoop to offer data storage scalability and on Spark for high-performance analytics. *Xurmo* can act as a dedicated and fast data mart developed for investigative analytics.

Xurmo is designed specifically for investigative analytics.



02

The Need for Investigative Analytics

The Evolving World of Reporting and Analytics

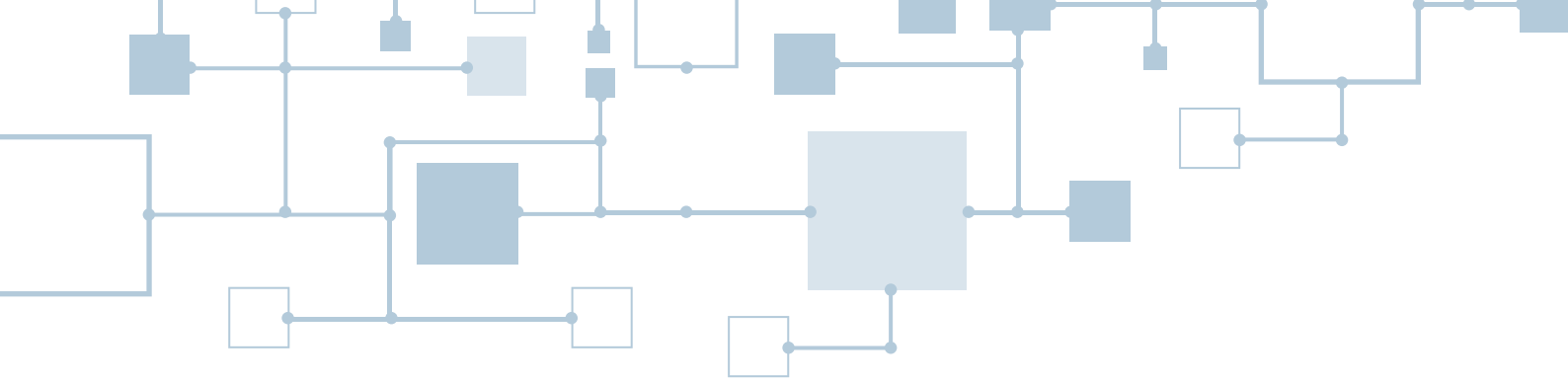
The analytical and reporting demands of managers and decision makers have changed, are changing, and will continue to change. At first, users were satisfied when they could run simple reports that would answer questions such as “Get the total amount of sales per region.” These reports were normally pre-defined. Users couldn’t develop new reports themselves, but these were developed for them by specialists in the IT department.

Since then, user reporting requirements have evolved. Nowadays, users request more self-service reporting capabilities so that they can create and change reports themselves. They also need more advanced analytical and statistical capabilities. For example, they want to run highly complex statistical models that show the likelihood that certain customers will switch to a competitor.

Investigative Analytics

But this is not where the story ends. A form of analytics that has definitely raised the bar for reporting and analytical tools is *investigative analytics* or *data discovery*. With investigative analytics users are free to analyze the data the way they want. Users are not limited in their discovery capabilities by the data structures of the tables of the pre-defined relationships between tables. Real investigative analytics makes it easy to analyze data across predefined as well as non-defined relationships.

With investigative analytics data can be analyzed across non-defined relationships.



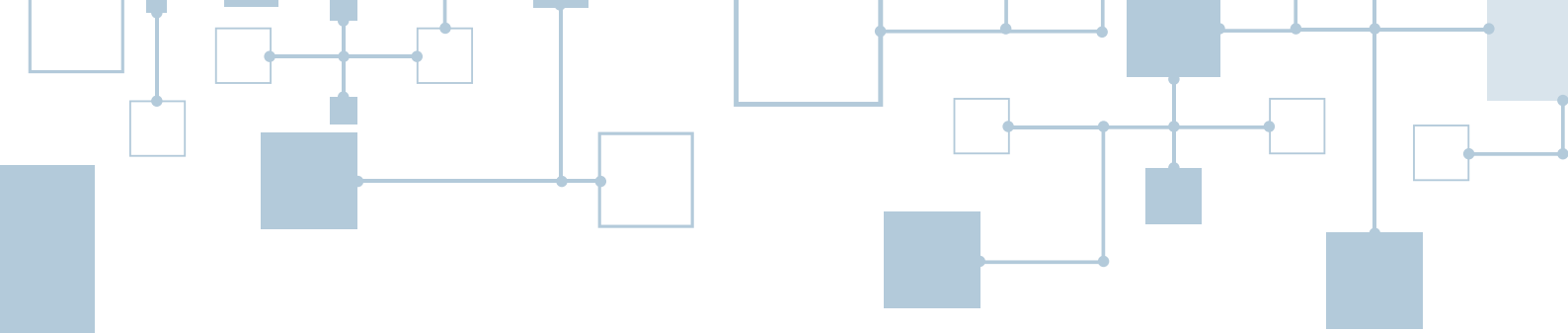
Let's describe an example that requires investigative analytics. Imagine a manager responsible for a number of retail stores in the Boston area who is aware that there is a problem with the sales of all types of sodas in his stores: sales are not as high as planned. It's important for him to find out what the reason is, which may not be that evident. In fact, it could be anything. The reason could be that lately shipments to the Boston stores have experienced serious delays, which means that certain products are not in the store on time. Or, it could be that the number of employees present in his stores is very low due to some contagious form of the flu. The effect is that the stores operate too slow, shells are not refilled, and the unloading of trucks is taking too long. All this can impact sales negatively. There could even be an external reason. For example, the weather may be so bad that customers are postponing their shopping, or a competitor has started a special promotion to sell sodas for half the price.

The manager's question is: What's causing the drop in soda sales? Is it the late shipments, overall sickness, the bad weather, the competitor's special promotion, or something else?

Where Other Tools Fall Short

Classic reporting tools won't help to find the source of this problem. Reports developed with such tools can show that the sales are behind, but they won't tell the reason why. These classic tools are restricted to showing pre-defined reports, and if no report has been developed to answer this particular question, the manager won't find the source of the problem.

Analytical tools offer the store manager features to look at 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. For example, if in a multi-dimensional cube, sales data is not related to delivery data or to employee sickness, it won't be able to show that it's the delayed delivery that's causing all the problems.



Statistical and data mining tools won't help either, because these tools, although very powerful, can only create models on data that has been selected. In a way, these tools have to be guided. If the manager knows with some certainty that it's the delivery that's causing the problem, he could use tools to confirm statistically that this is indeed the case. But the manager doesn't know this yet, so he is still looking for the reason. And, as indicated, the reason can be anything.

The manager needs a tool that allows him to query, browse, and analyze data without any restrictions. It should allow him to relate data elements for which no relationships or tables have been pre-defined. He must be able to ask questions such as "What's happening in Boston?" or "Is anything special happening in the week of September 15?" And if an answer is returned, he should be able to continue that path of investigation. That is investigative analytics.

Investigative Analytics and Urgency

Investigative tools are most often deployed in an operational environment, such as the one described. In such situations, a solution to the problem must be found urgently. Every second or minute of delay can cost the organization money or a business opportunity is missed. *Urgency* is a key characteristic for most of the problems solved with investigative tools.

Summary

Tools for investigative analytics allow users to freely query data without any form of restriction. These tools don't replace existing business intelligence tools, but enrich the palette of existing reporting and analytical capabilities; they fill a gap.

03

Requirements for Investigative Analytics

Tools for investigative analytics must meet the following requirements.

Free Form Querying

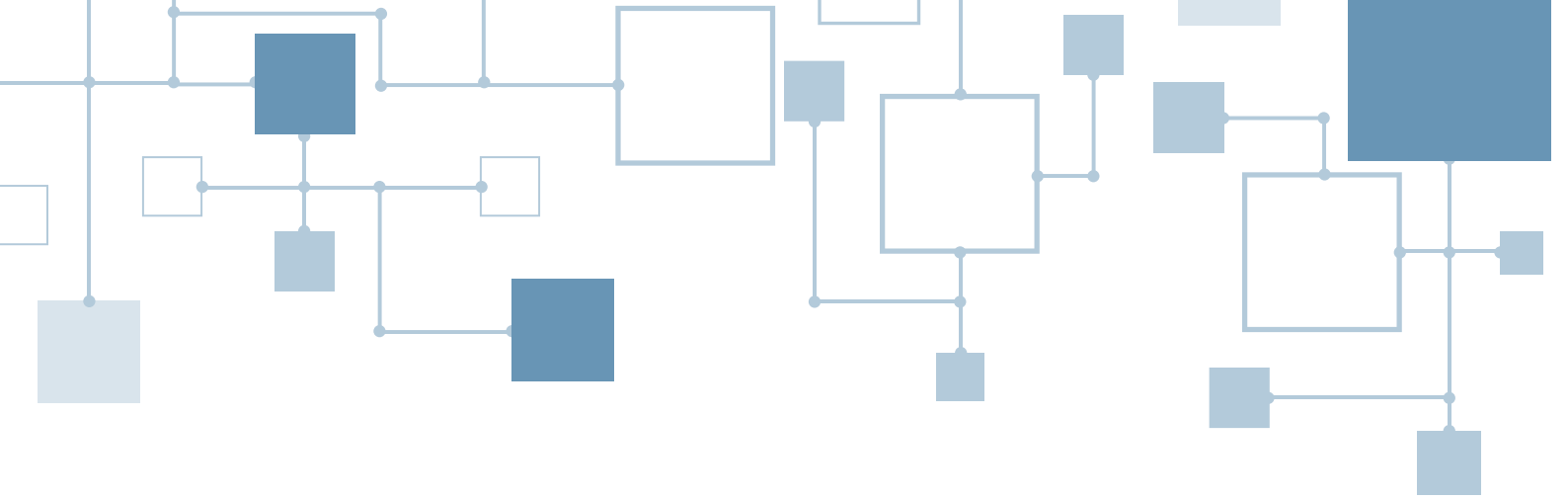
For investigative analytics users need to freely browse and analyze all the available data. They need to relate data elements that have not been associated before. In other words, they must be able to discover unknown relationships.

Any Data Source

It's convenient for every form of reporting and analytics when all the data needed is stored in one SQL database,

A tool for investigative analytics must be able to access any kind of data source.

such as a data warehouse or data mart. Nowadays, data is stored across all kinds of files and storage systems, such as Excel files, CSV files, documents with JSON structures, Hadoop files, NoSQL databases, and so on. To make all this data available in a data warehouse, dedicated programs must be developed that copy data from the original data sources to a SQL-based data warehouse or data mart. That's a time-consuming development project. All the while, the data can't be used for investigative analytics. A tool for investigative analytics must be able to access any kind of data sources directly and easily.



Instant Data Integration

It must be easy to make a new data source available for analytics. It should not be necessary to study the structure and the contents of the data source first and then write complex logic to make it available. The tool should be intelligent enough to load any new data source and to integrate it with the existing data instantly. This requirement relates directly to the need for urgency.

A tool for investigative analytics must be able to integrate data sources instantly.

Advanced Analytical Features

To be able to freely access and select data and to be able to look at the data values itself, is very valuable. But it's not always enough. After selecting the data, it may be necessary to use reporting and analytical techniques to study the data.

For example, to present the selected data as a chart can be quite insightful. Or, the answer to a user's question may become visible by applying analytical techniques, such as clustering, forecasting, or classification. So, a tool for investigative analytics must support advanced forms of analytics and data mining.



High Performance

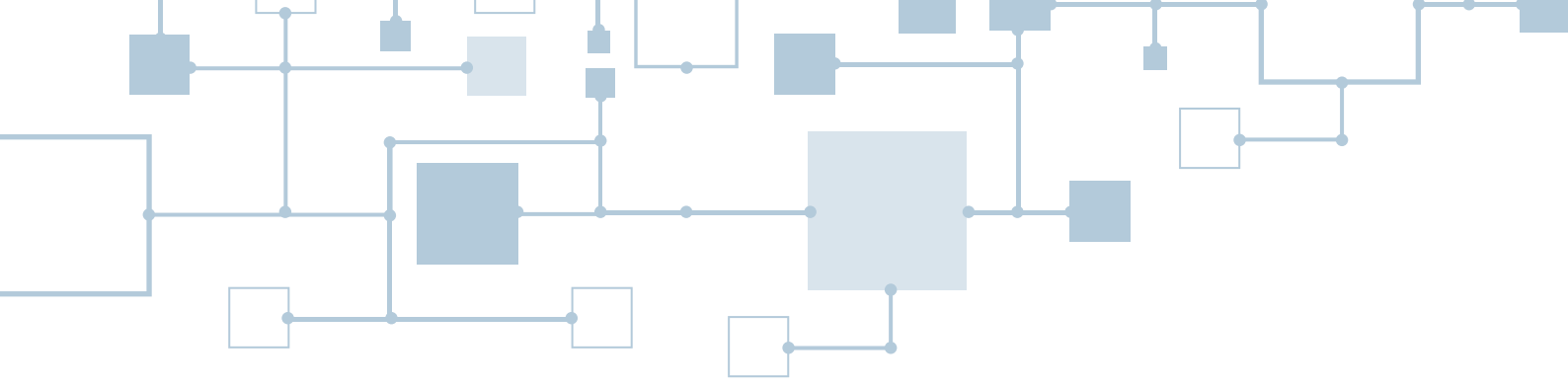
Investigative analytics is always an iterative process. At the beginning, users don't know exactly what they are looking for or what their question is. Usually, they deploy an iterative approach that brings them closer to the answer with each step. This implies that the user asks a question, receives an answer, creates the next question based on that result, receives a new answer, and so on. If the processing of each question takes minutes (or even hours), it would make a user hesitate and stop him from investigating certain options. It slows down the thinking process too much. So, high performance is crucial for tools supporting investigative analytics.

Investigative analytics demands high query performance.

Big Data

Two reasons exist why for investigative analytics must allow users to analyze massive amounts of data. First, due to their size, many new data sources can be qualified as big data sources, such as those containing weblog records, call detail records, sensor data, and call center transcripts. Investigating such big data sources can lead to valuable insights (see also the next section). Second, in many BI systems, users don't access the data warehouse, but a data mart. Commonly, data marts do not contain data on the lowest level of detail, but somewhat aggregated data. Investigative analytics requires data to be available on the lowest level of data, because for some queries the answer may only become visible when the lowest level is shown. The amount of detailed data can be substantial.

Investigative analytics must allow users to analyze big data.



Correctness of Results

Tools for investigative analytics must allow users to process correct queries that don't return 100% perfect results with respect to the original question. Imagine a user needs to combine customer data coming from an SAP system with customer data contained in an internally-developed system. Imagine also that no keys exist on which the customers can be joined. In that case, a "soft"-join must be executed. The result is probably close to correct, but not 100%. Still, the result could be very useful for the users.

A 100% perfect result can probably be implemented by starting an integration project. In this project, ETL programs are developed that use complex algorithms to integrate, cleanse, and standardize the data from the two systems. In addition, a dedicated database is developed in which the integrated customer data is stored. When that database is queried, the result is probably perfect. However, such a project may take months, and the urgency of the business problem may not allow for this delay.

If an urgent need exists to get a quick answer, maybe a less-than-100% result is good enough. What's more important: a perfect answer that is too late or a close-to-perfect answer that is on time? In many situations, users opt for the latter.

04

The Ever-Growing Mountain of Data

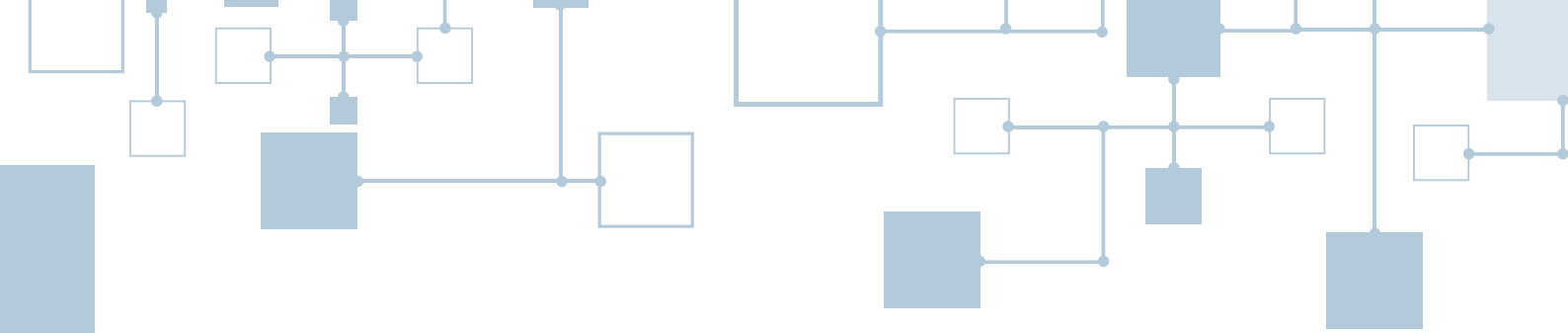
In general, there is a strong relationship between *data availability* and analytics, and definitely investigative analytics. The more data available, potentially the more extensive the reporting and analytical capabilities.

The Big Data Era

There was a time when all the data used for reporting and analytics was coming from an organization's own internal systems. Currently, we are clearly living in the *big data* era. Massive amounts of data are available on every aspect of the business. Customer data is available in sales systems and CRM systems, weblog data exists that show customer behavior on the website, social media networks contain data on customers, socio-demographic data on customers can be acquired, and the list goes on. Potentially, organizations can get a full picture of who their customers are. And all this data doesn't only tell an organization which and how many products they have ordered, but also shows also what they like, what their relationships are with other customers, what their hobbies and interests are, the neighborhoods they live in, and so on.

We are living
in the big data era.

But it's not only the availability of data on customers that has increased, it's on all aspects of a business. For example, there is abundant data available to support transport departments, such as data on traffic situations, on weather conditions, and on the state of the trucks. There is even data available to analyze whether drivers really use the most efficient routes to deliver parcels.



The machines in factories are overloaded with sensors to determine how well the machines are doing, or whether a technical malfunction is about to happen. On every aspect there is a mountain of data available: human resources, finance, logistics, procurement, production, and so on.

The Internet of Things (IoT) is going to generate even more data. More and more devices and machines, such as thermostats, cars, lights, alarms, and even shoe insoles, are hooked up to the internet and generate continuous streams of data. Analyzing all this data can lead to very valuable business insights. The IoT lets organizations evolve from data-poor to data-rich organizations.

The Data Warehouse

The classic approach to make all the data available for analytics and reporting is to copy it to a centralized

Is the popular data warehouse the right data integration solution for investigative analytics?

environment first: *the data warehouse*. In the data warehouse all the data from many data sources is integrated. This sounds simple and straightforward, but copying data to the data warehouse and integrating all the data is most often a complex and resource intensive exercise. Tables must be designed and developed in the data warehouse to hold the new data; ETL programs must be developed to extract data from the data sources and to transform and load the data into the data warehouse; and, the ETL programs must be scheduled and the entire process must be managed. Especially developing the logic to transform the data so that it “fits” in the data warehouse can be time-consuming.

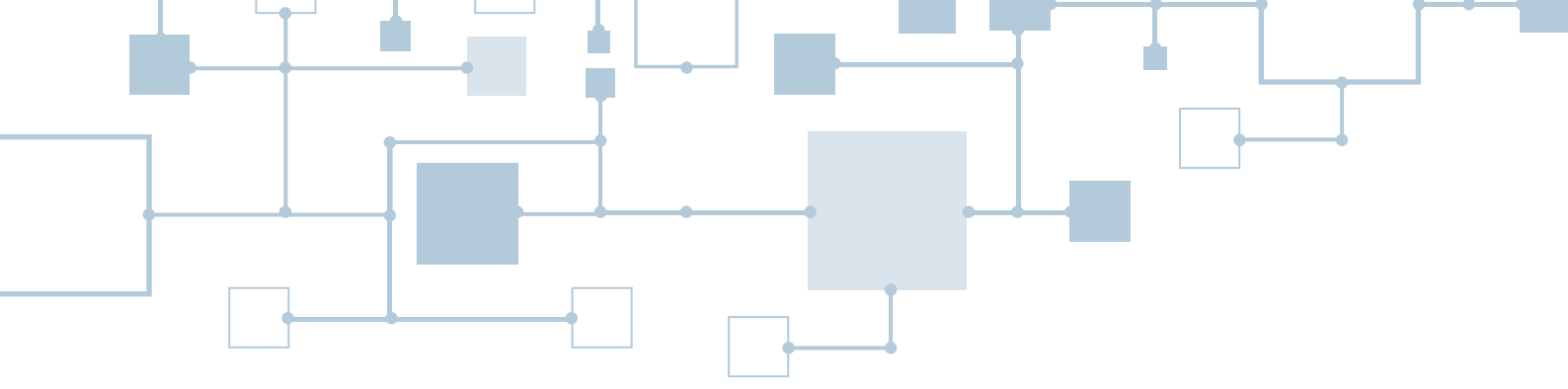


Many reasons exist for copying all the data to the centralized data environment first and to process it before making it available. First, by placing all the data in one database, developing reports that access data from multiple sources is easy. Second, by transforming, standardizing, and cleansing data, the quality of the report results improves. Third, querying the data sources directly may be slow or may cause technical problems, such as interference or concurrency issues.

The Need for Instant Data Integration

The consequence of working with a centralized data warehouse is that data from a new source becomes only available for reporting after a number of weeks or even months, because developing an integration solution can take quite some time. For particular forms of reporting and analytics, such as financial reports that are executed repeatedly, this is more than acceptable. In fact, it's most likely a requirement that the report results are 100% correct, and thus the integration solution must be professionally developed and extensively tested.

On the one hand, a centralized integration solution is ideal for investigative analytics, because the data warehouse contains a wealth of data, making it a valuable source for investigation. On the other hand, it's not ideal when a new data source needs to be analyzed instantly. If an analyst needs to investigate a new data source plus existing data, the integrated result must be available in seconds and not in weeks. Therefore, tools for investigative analytics must support instant data integration.



100% Correct Results

Analyzing a source of which the data hasn't been studied, hasn't been cleansed, and hasn't been synchronized with the existing data, can lead to imperfect results. For example, when a customer appears in the new data source and in the data warehouse, its company name may have been spelled differently in the two systems. How does the user know that the two records represent the same company? Such a situation can skew the reporting results somewhat.

But is 100% correctness always important? In some investigative exercises speed may be more important than 100% correctness. This is especially true for those situations for which an answer must be found urgently. The need for 100% correct report results is desirable, but not always required for investigative analytics. If a user needs a new data source, it must be possible to analyze the data instantly, even if that lowers the correctness of the result somewhat.

05

Overview of Current Data Store Technologies

To make data available for investigative analytics, it must be stored in some data store. The data store technology used can enrich or limit investigative analytics. For example, if the data store technology only allows tables to be combined via predefined relationships, it clearly sets limitations, because new and undefined relationships can't be analyzed. Another limitation would be if the technology doesn't allow analysis of textual data, such as email messages, social media messages, and call center transcripts.

Current data store technologies can be divided in three main groups: SQL, NoSQL, and SQL-on-Hadoop. This section describes how well they support investigative analytics respectively.

05.1

SQL as Data Store Technology for Investigative Analytics

For a long time, SQL database servers, such as IBM DB2, Microsoft SQL Server, and Oracle, have been used for storing all the enterprise data. They are robust and feature-rich products and have proven their worth over and over again. SQL itself is a language that can be used for reporting and supports analytics.

SQL products have their limitations when used for investigative analytics.

From the standpoint of investigative analytics, SQL has three limitations:

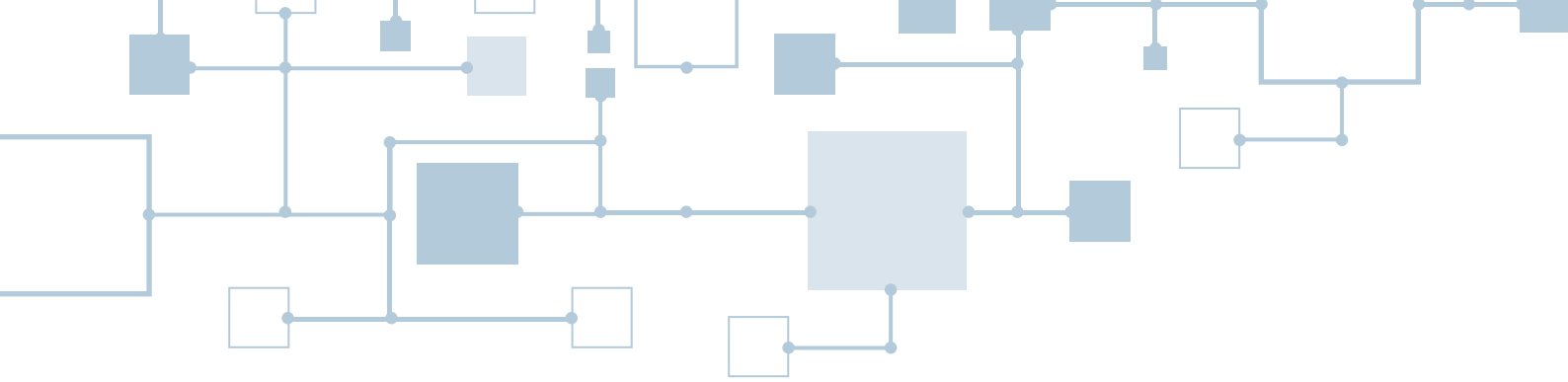
- Users must know the data structures of tables and columns.
- Users must know the undefined relationships between tables.
- Users work with static queries.

Data Structures Must Be Known

When users write SQL statements, they must know in which columns and tables the data is stored. For example, if a user wants to analyze customers based in Boston, he must know in which column he can find the customer's city. The following query wouldn't work in SQL:

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

This query suggests that SQL must look for all the records in the CUSTOMERS table in which one of the columns holds the value Boston. Today, this is not supported by the SQL products.



It's even more difficult if users don't know exactly in which table the requested data is stored. The following query can't be processed either, because a FROM clause, indicating the table to be accessed, must be specified:

```
SELECT *  
WHERE * = 'Boston'
```

In SQL, users must specify precisely which tables to query to look for the value Boston. The query would look somewhat like this:

```
SELECT 'CUSTOMER', CUSTOMER_ID  
FROM CUSTOMERS  
WHERE ADDRESS_CITY = 'Boston'  
UNION  
SELECT 'INVOICE', INVOICE_ID  
FROM INVOICES  
WHERE DELIVERY_CITY = 'Boston'  
UNION ...
```

For each table containing city names, a comparable query must be added to this statement. A time-consuming task, and as indicated, it requires that users know exactly which columns contain city names. In addition, every time a new data source is added that contains city names, the above query must be extended.



Undefined Relationships Must Be Known

The other restriction is related to the relationships between tables. SQL allows tables to be combined in any possible way. But the user must know on which columns it makes sense to combine tables. For example, it probably never makes sense to join the customer table with the product table based on respectively the name column and the price column, like this:

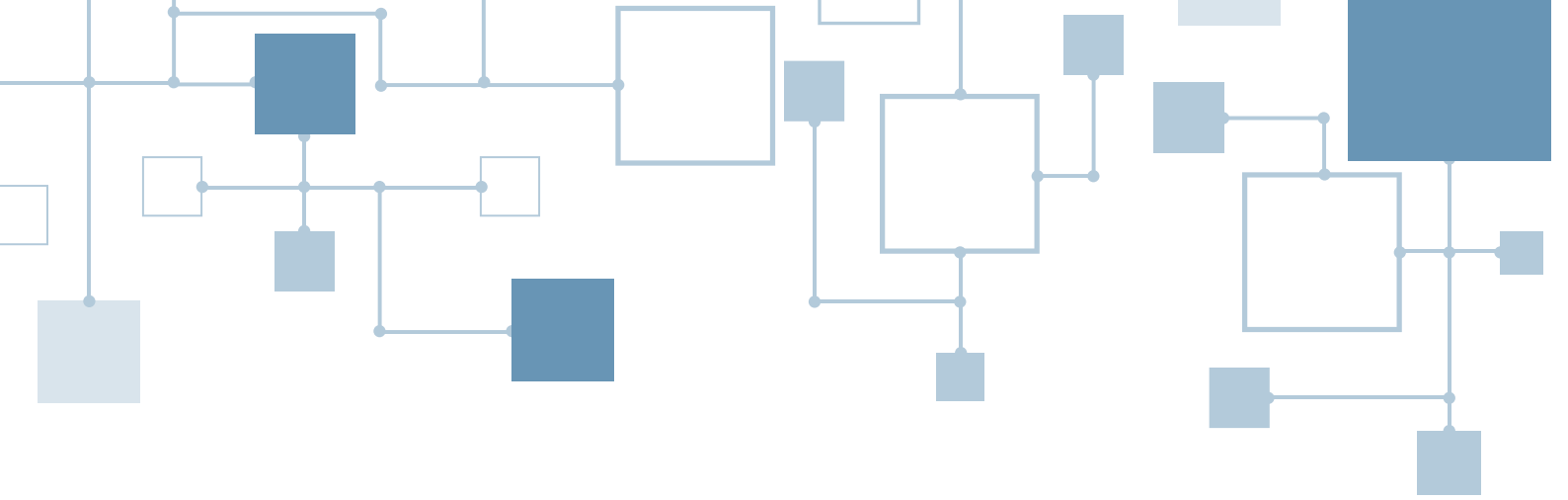
```
SELECT *
FROM   CUSTOMERS INNER JOIN PRODUCTS
ON     CUSTOMERS.NAME = PRODUCTS.PRICE
```

Still, SQL allows it.

But which relationships do make sense? If a foreign key between two tables has been defined, every user knows that a relationship exists between the columns. But many more relationships can exist and can be useful for investigative analytics. These relationships are not defined in SQL database servers and are therefore called *undefined relationships*. For example, if the customer table has two columns containing city names, such as the default delivery city and the address of headquarters, a relationship does exist. For users it may be worthwhile to study this relationship. Another undefined relationship exists between the country column in the deliveries table and the country column in the warehouse table.

How to find undefined relationships with SQL?

If the only table a user wants to analyze is the customer table, he should be able to find all the undefined relationships just by looking at the data itself. But when his database contains over a thousand tables and tens of thousands of columns, it's impossible for a user to know all the undefined relationships and what kind of values all the columns contain.



Static Queries

When new data is added to an existing SQL database, new tables must be developed and new columns must be added to existing tables. SQL supports statements to do that, such as the ALTER TABLE statement. However, if new data is added, the existing queries and reports are not automatically changed accordingly. If a query accesses two tables with customer data, it's not automatically extended when a third table with customer data is added. All the relevant queries must be changed by hand. So the programmed queries are all static, they are not dynamic in the way that they are adapted when the data structures change.

Summary

Current SQL database servers are not ideal for investigative analytics when the user wants to freely access the data, when he doesn't want to be restricted by the data structures, and when he doesn't have to be aware of all the undefined relationships.

05.2

NoSQL as Data Store Technology for Investigative Analytics

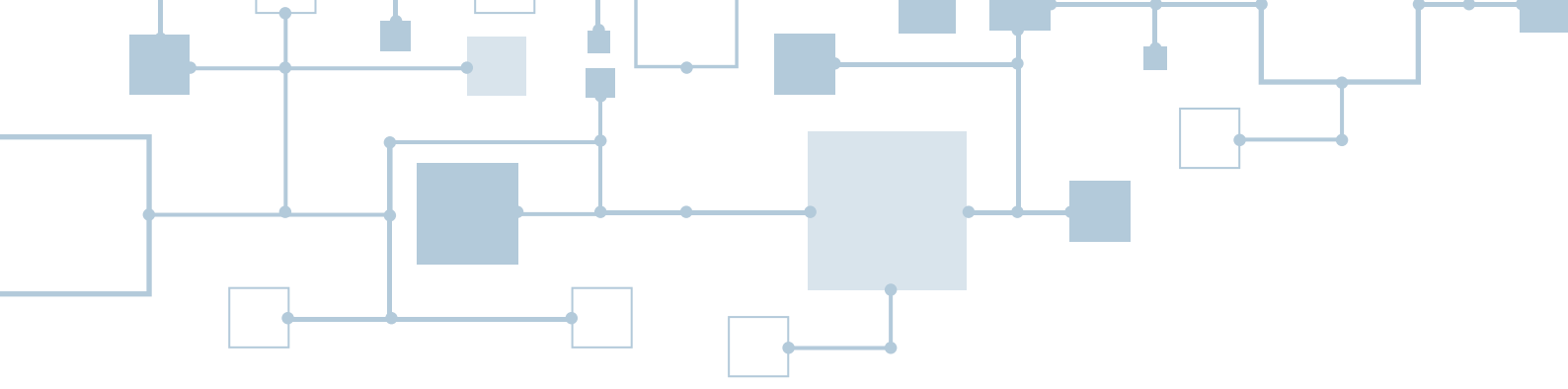
NoSQL products, such as MongoDB, Apache HBase, and Cassandra, are representatives of the new generation of data store technologies. This section describes whether the same limitations for investigative analytics that apply to SQL, also apply to the NoSQL products.

Data Structures Must Be Known

In NoSQL, data isn't always organized in flat, relational data structures. For example, several NoSQL products use JSON or BSON structures to store data with a hierarchical structure. In these systems a file consists of documents, and each document is organized as a set of elements with hierarchical structures. So, data that logically belongs together can be stored together. Each document in a file can have a different structure. From a data modeling standpoint, NoSQL products are more flexible.

In NoSQL, data isn't always organized in flat, relational data structures.

Still, the data stored in NoSQL products does have a structure. Therefore, users who write queries for NoSQL products must understand these data structures. They must know in which elements data is organized and what the hierarchical structures are. This means that equivalent versions of the SQL statements in the previous section, such as `SELECT * WHERE * = 'Boston'`, are not possible with NoSQL either. Even in NoSQL, users must specify explicitly which files/tables and which elements to access. This is not very accommodating for investigative analytics.



Undefined Relationships Must Be Known

In most NoSQL products no relationships can be defined, not even foreign key equivalents. This means that users must know which of the elements can be combined and how data from multiple tables can be joined. They must know the undefined relationships.

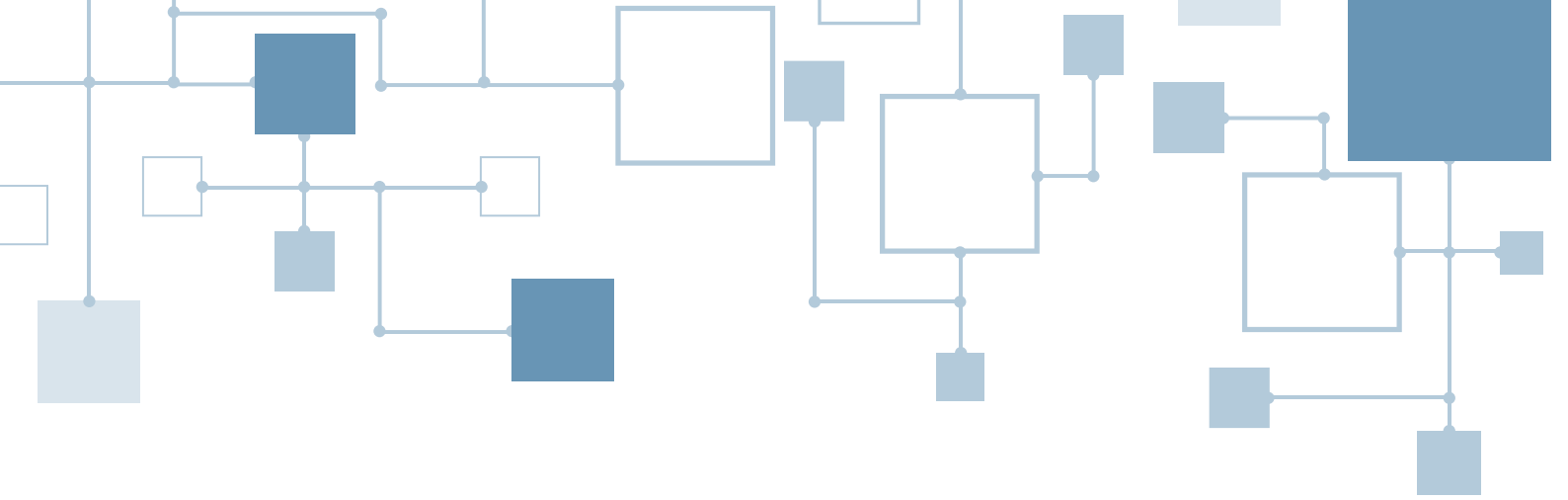
What applies to the SQL products, applies for the NoSQL products as well: it makes no sense to join the customer table with the product table based on respectively the name column and the price column. In fact, most NoSQL products do not even supports joins.

Static Queries

Most NoSQL products support variable data structures which means that data structures can be changed on the spot. For example, if a new record has one element more than the records stored, that new column is added on the spot when the record is inserted. It's a big benefit that data structures can be changed so easily, and that different records in the same file can have different structures, but when a data structure is changed, the queries are not automatically updated. This is a manual process.

Summary

Undoubtedly, NoSQL products have their strengths, such as transactional speed, being able to handle big data, and dealing with complex data structures. But their strength is not in supporting investigative analytics.



Additional Remarks

From the perspective of investigative analytics, here are some extra remarks with respect to NoSQL:

- NoSQL products support very simplistic query languages. They are far from the query richness SQL offers. This means that users must develop complex programs to analyze the data.
- The languages supported to query the data are too technical for business analysts and business users. Usually, in-depth knowledge of Java or a similar programming language is required.
- Dedicated tools exist for reporting and analytics that operate directly on Hadoop and NoSQL products and hide these technical interfaces. Unfortunately, these tools don't support investigative analytics. Most of them offer a familiar form of reporting or analytics. In addition, they can only access one data source. In most cases this is Hadoop. They don't support reporting over multiple different data store technologies.

05.3

SQL-on-Hadoop as Data Store Technology for Investigative Analytics

SQL-on-Hadoop and Schema-On-Read

The third option is *SQL-on-Hadoop*. A SQL-on-Hadoop engine is a SQL engine that makes use of the Hadoop framework and stores its tables in HDFS files. Most of these

engines support a relatively rich SQL dialect. Examples of such products are Apache Hive, Apache Drill, Cloudera Impala, and Spark SQL. Some of these engines are new and have specifically been developed to access Hadoop, while others are ports of existing SQL servers to Hadoop.

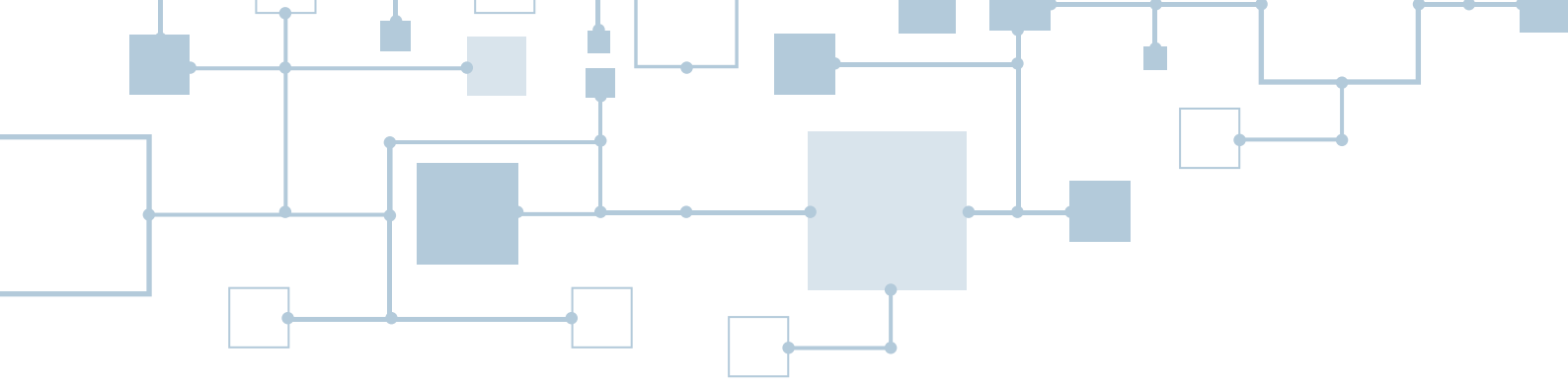
Classic SQL systems support the concept called *schema-on-write*. Schema-on-write means that all the data written to a database has a schema. For each value of each record it is known to which column of the table it belongs and what its data type is. A schema is not optional in classic SQL. The opposite of schema-on-write is *schema-on-read*; when a database server uses schema-on-read the stored data has no schema. Or, more precisely, the database server doesn't know what the schema is, the data values are like blobs of bytes. Because tools for reporting and analytics can't handle schema-less data, the data read from the database must have a schema. Assigning a schema happens when the data is read, hence the name schema-on-read.

The key benefit of schema-on-read is *flexibility*. If new data with a different structure is added, there is no need to change the database structure, no need to reorganize all the stored data, and no need to unload and reload all the data. It's just that the

The key benefit of schema-on-read is flexibility.

schema-on-read logic must be changed. Another benefit is that different tools can assign a different schema to the same data and thus work with a different view on that same data.

Although SQL-on-Hadoop engines are usually associated with the concept of *schema-on-read*, most of them don't support it. Therefore, this group of engines is as useful for investigative analytics as classic SQL products; see Section 5.1.



Data Structures Must Be Known

When users use SQL-on-Hadoop engines that do support schema-on-read, they still have to indicate from which tables and columns data must be retrieved. They still have to understand the structure of the data. The fact that schema-on-read is used under the hood to transform the schema-less data to schema-rich data, doesn't influence that. The SQL user experiences a schema-rich data environment with all its pros and cons.

Undefined Relationships Must Be Known

Most of the SQL-on-Hadoop engines do not support primary keys and foreign keys. So, no relationships can be defined; all the relationships are undefined. There is no mechanism in the SQL-on-Hadoop engines to make the undefined relationships explicit.

Static Queries

The schema-on-read concept allows for very dynamic data structures. New columns can be added easily to existing tables. But if new data is added, the existing queries and reports are not automatically updated. If a query accesses two tables with customer data, it's not automatically extended when a third table with customer data is added. All the relevant queries must be changed by hand.

Summary

What's special about some of the SQL-on-Hadoop engines is that they support schema-on-read, which, from a data management perspective makes the environment very flexible, but SQL-on-Hadoop by itself doesn't solve the challenges of investigative analytics. More functionality is required.



06 | Xurmo Explained

This section describes *Xurmo*, a tool designed specifically for investigative analytics.

To be able to handle big data, Xurmo's database technology makes use of one of the most scalable data storage platforms: *Apache Hadoop*. To process its queries fast, it's fully integrated with *Apache Spark*. Note that both Apache modules come as part of Xurmo, and don't have to be acquired or installed separately.

To understand how Xurmo implements free-form queries and how it supports investigative analytics, the next subsection describes some of the Xurmo concepts. Section 6.2 describes Xurmo's query capabilities and Section 6.3 its internal architecture.

Xurmo makes use of
Apache Hadoop and
Apache Spark.

06.1

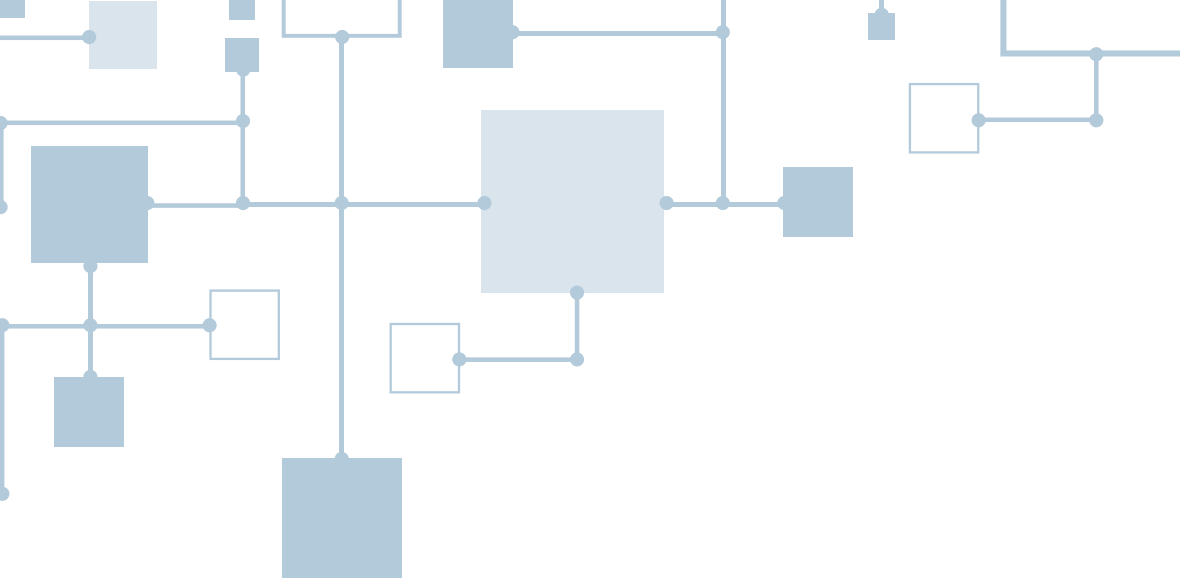
Xurmo's Unique Concepts

Xurmo is not really a SQL database server, it's not a traditional SQL-on-Hadoop engine, nor is it a NoSQL database. Section 5 describes the limitations of these technologies for investigative analytics. To be able to support investigative analytics, to deal with dynamic data structures, and to offer free form queries, Xurmo comes with its own unique approach to organizing data.

In Xurmo tables don't have to be designed upfront, the data types of columns don't have to be selected, nor do relationships have to be defined. Raw data from all types of sources can be loaded without any development work upfront. No ETL logic has to be written on forehand. If users want to analyze a new data source, the only thing they have to do is connect Xurmo to the data source, and Xurmo loads the raw data that becomes instantly available for investigative analytics.

In Xurmo tables
don't have to be
designed upfront.

To make this possible, Xurmo distinguishes between *data structure* and *meaning*. Data structure refers to the Cartesian co-ordinates of every data value in such a way that the relative position of each data value from the other is determinable. Imagine that the data values "London" and "Tokyo" appear in the 5th column of a table in rows 3 and 7 respectively. Their positions indicate possible relationships between each other. If they appear in the same column there exists a relationship (same type of value), and if they appear in the row there exists a relationship as well (these values describe characteristics of the same entity).



The meaning of data is a use-case specific, human interpretation of the relative positions of data values based on the co-ordinates of two or more data elements. In the example above, the human interpretation of the fact that “London” and “Tokyo” appear in the same column can be that they are both capital cities or both country head offices of companies.

When raw data is loaded, Xurmo constructs a Cartesian space in which each data value gets a unique coordinate. Next, data can be searched by triangulating across their row, column, and source addresses along with time. The relative position of the data values are computed in runtime and applied to pattern discovery algorithms. By enabling search for data values based on their co-ordinates and allowing computation of patterns across data elements, Xurmo allows users to define and derive meaning from raw data instantly.

Note: Xurmo can load data from a long list of data sources and data formats, including XML, JSON, RSS, HTML, social media apps (like Twitter, FB, and LinkedIn), images, log files (machine data), MS Office applications (Word, Excel, PPT), PDFs, and SQL database servers.

06.2

Xurmo's Query Capabilities

Free Form Queries

Xurmo organizes the data in such a way that a fully free form style of querying is possible, which makes it ideal for investigative analytics.

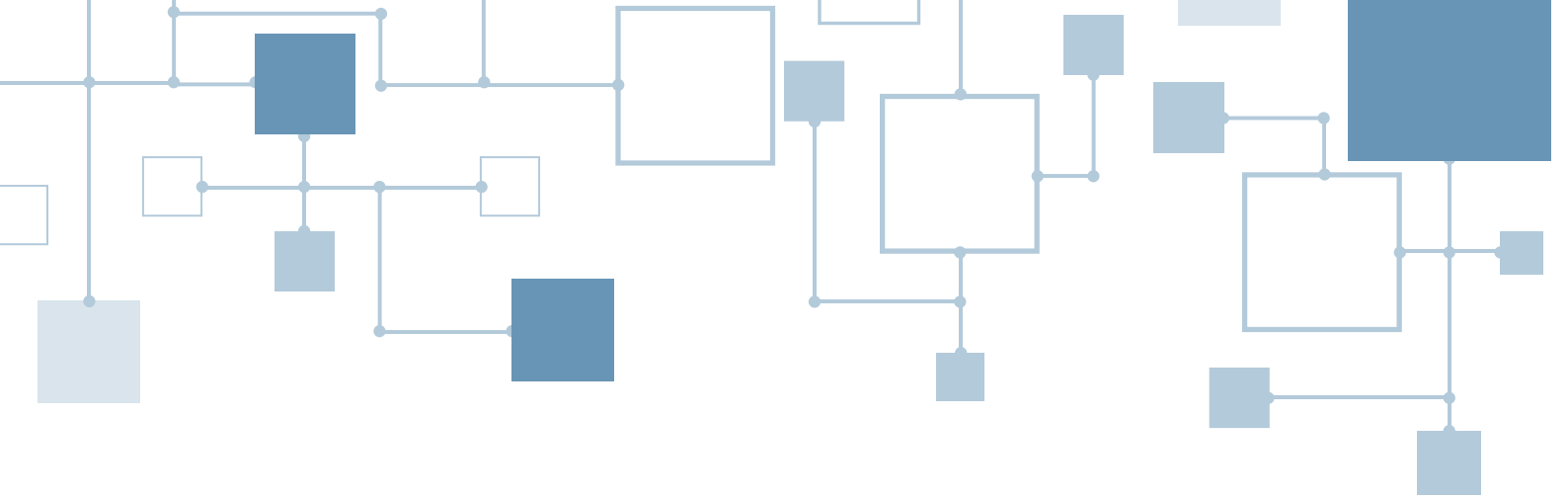
Xurmo supports a fully free form style of querying.

For example, the query to find all the customers from Boston is simple in Xurmo. Formulating and running queries is done with the Xurmo module called *TIQL*. To answer this query on Boston-based customers, the user only has to enter the word Boston. Automatically, TIQL lists all the data sources and columns in which the value Boston appears:

```
Source: CUSTOMERS Field: ADDRESS_CITY  
Source: CUSTOMERS Field: DELIVERY_CITY  
Source: DELIVERIES Field: CITY
```

The user then picks a column and right away TIQL generates the following query:

```
SELECT CUSTOMERS.ADDRESS_CITY  
FROM CUSTOMERS  
WHERE CUSTOMERS.ADDRESS_CITY = 'Boston'
```



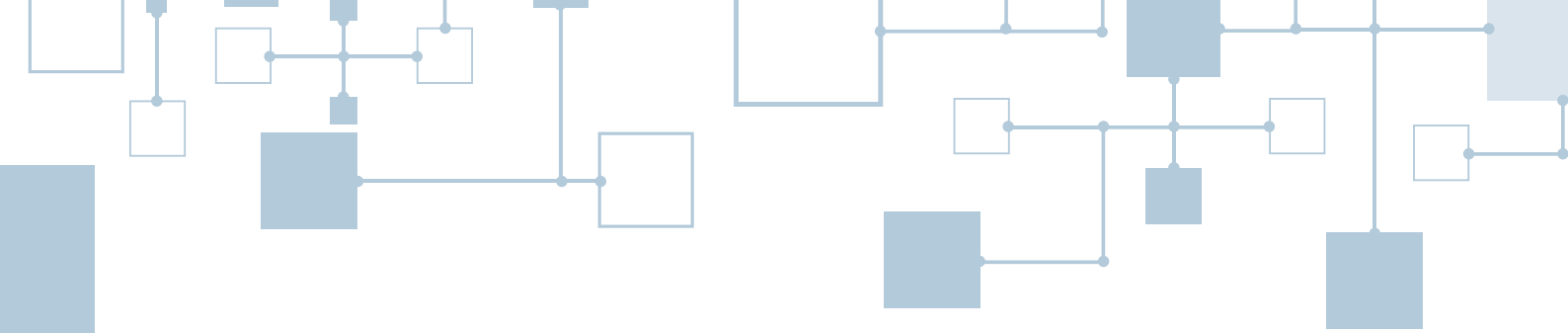
Users can add other columns and all kinds of filters in the same way. So, they don't have to know on forehand what the structure of the tables is, or in which columns city names such as Boston appear. This is especially relevant for new data sources that haven't been studied by them yet. After a new data source has been loaded, the users can start querying. In a way, by suggesting which column or table to access, *Xurmo guides* the users in developing their queries.

In Xurmo, links between words can be defined to enrich the querying capabilities. When a user looks for the word USA, it also finds the records where the names United States or United States of America are used, if these words have been defined as synonyms of each other.

Relationships are derived automatically by Xurmo based on the type of the columns, and the content of the columns. In addition, analysts and users can define their own relationships. This feature makes it possible to transform undefined relationships into defined relationships.

Xurmo uses these defined relationships when data is queried. For example, when users want to find the names of the sales people who have sold products in India, all they have to enter is India. Xurmo shows where that code appears:

```
Source: REGION Field: COUNTRY
```



The generated query becomes:

```
SELECT  REGION.COUNTRY
FROM    REGION
WHERE   REGION.COUNTRY = 'India'
```

Next, the user types NAME after COUNTRY in the SELECT-clause of the statement:

```
SELECT  REGION.COUNTRY, NAME
FROM    REGION
WHERE   REGION.COUNTRY = 'India'
```

Right away, Xurmo shows the tables in which the column NAME appears and with which REGION it has a relationship, which, in this example, is only SALES_PERSONS. When that table is selected, the query is expanded with a join:

```
SELECT  REGION.COUNTRY, SALES_PERSONS.NAME
FROM    REGION JOIN SALES_PERSONS ON REGION.REGION = SALES_PERSONS.REGION
WHERE   REGION.COUNTRY = 'India'
```

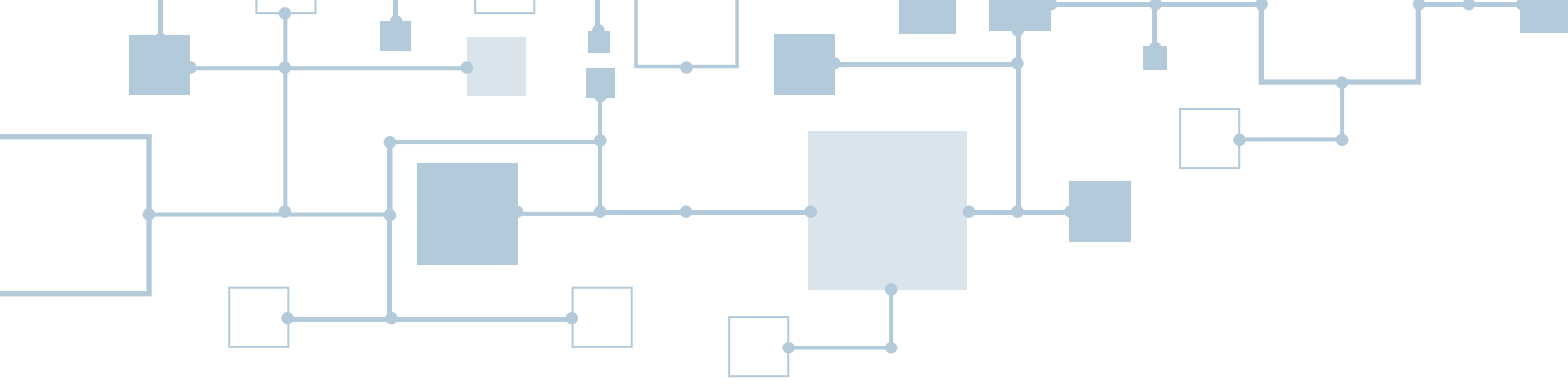
Another example is one in which there are two source tables with customer data. If a user wants to know the addresses of the customer called Metheny, the query is simply:

```
SELECT ADDRESS
WHERE  CUSTOMER_NAME = 'Metheny'
```

Xurmo locates all the customers in both source tables that have Metheny as name. If the user is interested in all the orders placed by those customers, the query is just slightly more difficult:

```
SELECT ORDER_NR, DELIVERY_DATE
WHERE  CUSTOMER_NAME = 'Metheny'
```

Xurmo uses the relationships from the two customer tables to the orders tables to come up with an answer.



Correctness of Queries

As shown, Xurmo assists and guides the users with developing queries. Besides guiding them, users can ask Xurmo for assistance in the form of statistics and recommendations.

Section 3 describes that 100% correctness of report results is not always required. Sometimes time is more important. Note that 100% correctness of results should not be confused with 100% correctness of the query. Xurmo helps users to develop 100% correct queries. The results will also be 100% correct with respect to the queries. It's just that the result may not exactly be correct with respect to the original questions of the users. Due to faulty data, such as misspellings and missing data, the result may not be perfect. But it's still correct with respect to the query. The assistance given by Xurmo helps users to become aware of the fact that the results may not be perfect. For example, users can see that some names are not spelled correctly, or that the number of different values in a column is not correct. It may lead them to reformulate the queries in such a way that they deal with the faulty data to improve the results.

Analytics

Data that has been selected can be saved and used for more advanced forms of analytics. Xurmo supports a wide range of analytical operations, including sampling, classification, regression, clustering. This allow users to create statistical models or rules. As example, Figure 1 contains the results of two analytical operations. Xurmo also allows the data to be processed by a model developed with the popular open source platform called *R*.

Xurmo supports a wide range of analytical operations.

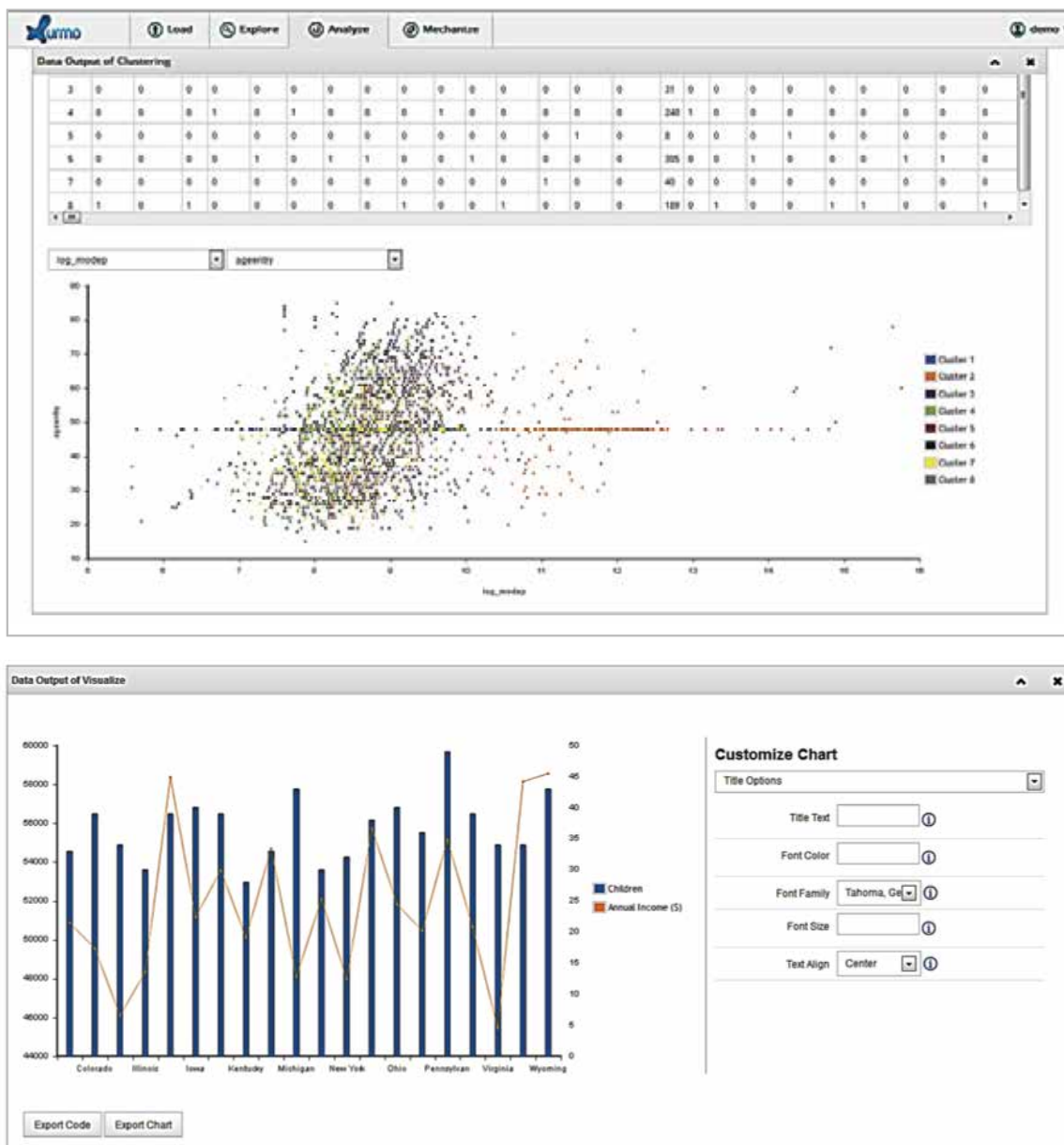


Figure 1 The results of two Xurmo analytical operations.

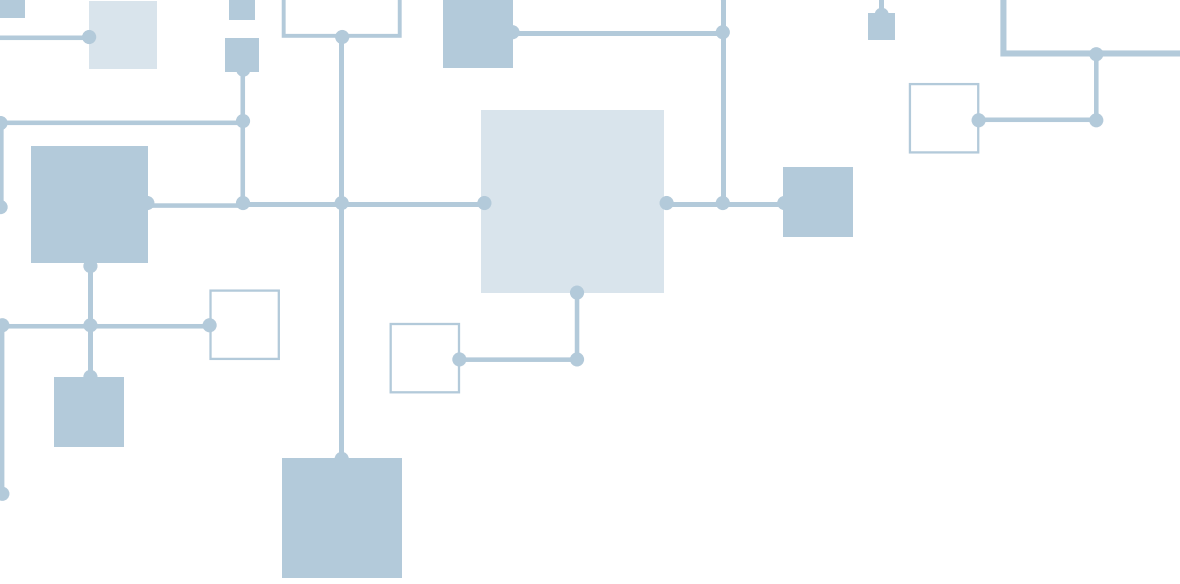


Embedding Analytical Models

Xurmo supports *embedding* of analytical models. With investigative analytics users try to find solutions to their business problems. The solution may be a set of records or a model if the analytical functions are used. In many situations, when a solution is found, that's where the exercise stops. To return to the example of the Boston store manager (see Section 3), if he discovers that the disappointing sales is primarily due to the flu, then temporary hands may have to be hired, or they just have to wait it out. Anyway, the reason for the urgent problem has been discovered, and the business can take action or not.

Analytical models created with Xurmo can be embedded in applications.

But this is not always the result of investigative analytics. It may be that after the rules or models have been discovered, they must be executed repeatedly. Imagine that a model is discovered to predict expected transport delays in the busy Boston area based on weather and traffic data. When such a model is discovered, it may have to be embedded in some application. With this application users can then enter a scheduled route, and the embedded analytical model predicts the expected delay. In this case, users don't have to start an investigative exercise over and over again. The *embedded analytical model* can be invoked endlessly.



Analytical Workflow

Besides support for embedded analytical models, Xurmo supports analytical models that consist of multiple analytical operations. For some decisions, it may not be as easy as to simply invoke one analytical operation, but it may be necessary to execute a series of them. For example, first a specific set of records must be selected, regression analysis must be applied, then the result must be joined with some external data, a sample must be created, then a segmentation algorithm must be applied, and finally, the result must be joined with some other data set. The result of one step can influence which next step is executed. In fact, an *analytical workflow* must be designed to get the right results. Xurmo allows for the definition and embedding of such analytical workflows. Figure 2 shows an example of such a workflow defined with Xurmo.

An analytical workflow consists of multiple analytical operations.

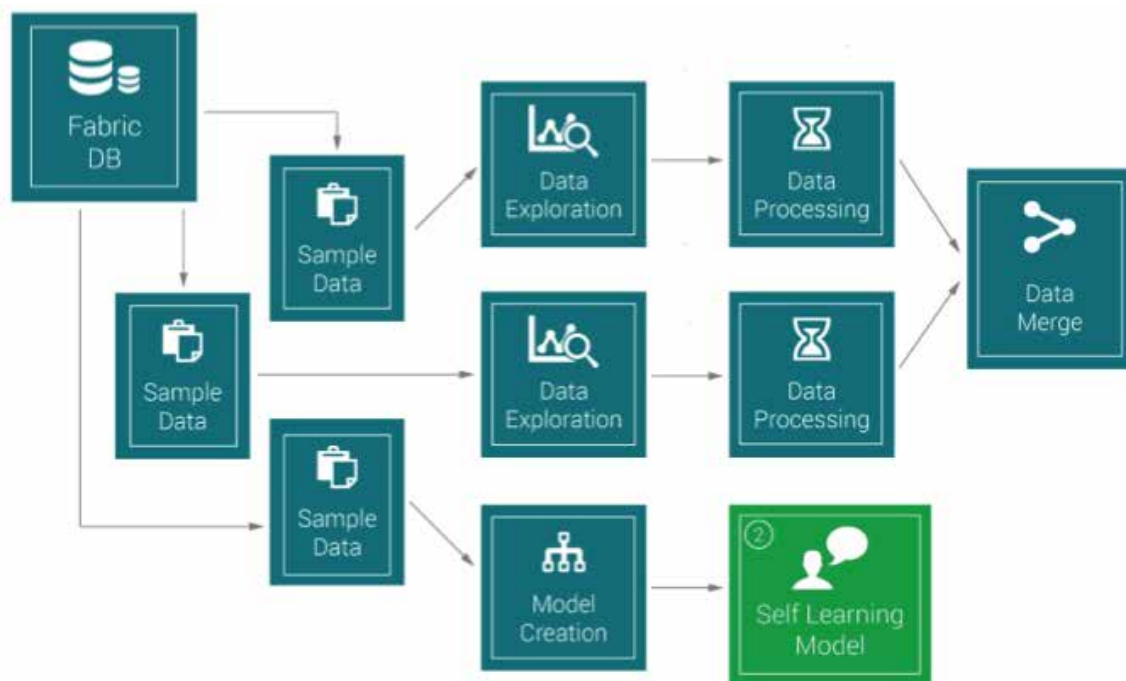


Figure 2 An example of an analytical workflow defined with Xurmo.

Self-Learning

Xurmo's platform enables the creation of self-learning models quickly and easily. Users can quickly build a model from a small seed training set, and then deploy it into production. Once in production, the training set is enhanced in two ways, through either user feedback or feedback from a rule-based classifier.

In the former case, users tag the data with what they think is the correct answer/class and this feedback is collated and summarized before being added to the training set. In the latter method, a set of rules is defined that allow the data to be classified with a degree of confidence. If the degree of confidence is higher than a predefined threshold, the data is added to the training set. In this way, the model is always being created from more and more data, allowing it to evolve into something that is more accurate and more in line with the current trends.

06.3 | Xurmo's Internal Architecture

Figure 4 presents a high-level overview of the internal architecture of Xurmo. The *Xurmo Data Loader* loads new data from data sources into Xurmo. The data loader stores all the data in Hadoop-HDFS files. So, when data is queried, HDFS files are being accessed and not the original data sources, allowing Xurmo to operate independently.

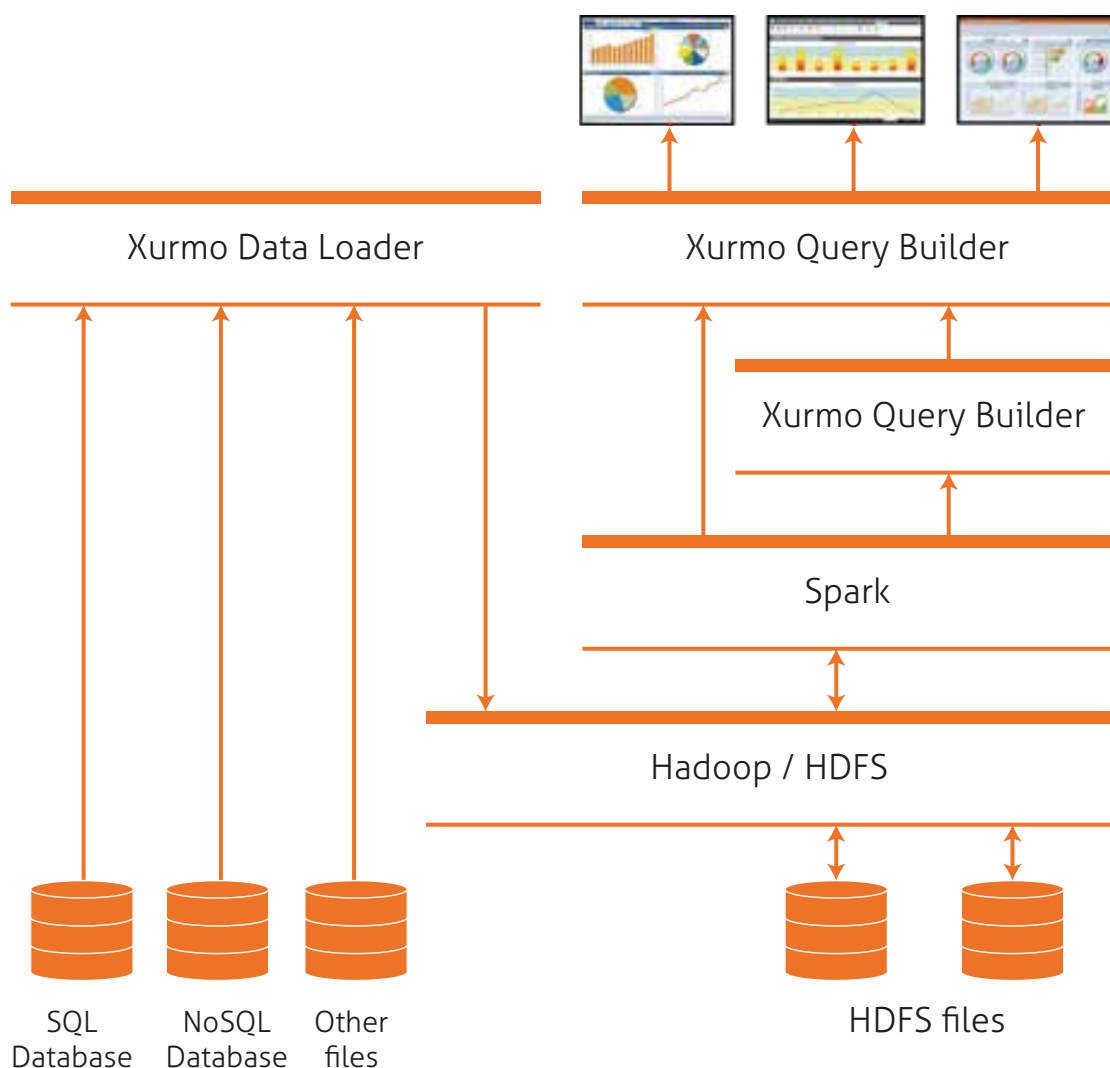


Figure 3 High-level Architecture of Xurmo.



Xurmo Exploits Apache Spark

Xurmo leans heavily on the open source parallel processing framework called *Apache Spark*. Spark became a top-level project of the Apache Software Foundation in February 2014. Version 1.0 of Apache Spark was released in May 2014.

The technology supports in-memory processing to boost the performance of big data analytics applications, but it can also do conventional disk-based processing when data sets are too large to fit into available system memory. Spark offers an application programming interface (API) and tools for managing and analyzing data, including a SQL query engine, a library of machine learning algorithms, a graph processing system, and streaming data processing software.

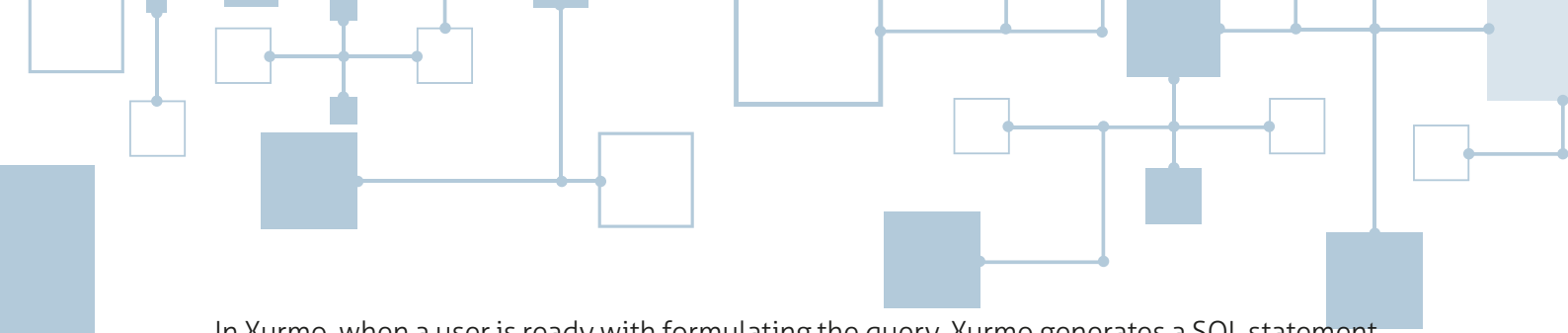
Xurmo uses the Spark open source parallel processing framework.

Spark provides programmers with a potentially faster and more flexible alternative to *MapReduce*. MapReduce offers a programming interface with which developers can write applications to query data in HDFS. MapReduce can efficiently distribute query processing over hundreds of nodes. Unfortunately, MapReduce has a batch-oriented style of query processing, making it not really suited for the iterative nature of investigative analytics.

Apache Spark can process data from a variety of data repositories, including the Hadoop Distributed File System (HDFS), NoSQL databases and relational data stores such as Apache Hive.

Xurmo Generates Spark SQL

Spark SQL is one of the most popular SQL-on-Hadoop engines. It fully exploits the in-memory power of Spark itself. With respect to the supported SQL dialect, it's compatible with Apache Hive.



In Xurmo, when a user is ready with formulating the query, Xurmo generates a SQL statement that is then executed by Spark SQL using Spark. Spark SQL doesn't execute these queries on the data sources, but on the copied tables residing in HDFS.

The fact that Xurmo fully exploits the data scalability of Hadoop, and the scalable in-memory query processing capabilities of Spark and Spark SQL together, make it a highly scalable tool for investigative analytics.

Note that Xurmo, in contrast to other analytical tools, can operate independently; it doesn't need a separate database server to store all the data. It comes embedded with the Spark and Spark SQL modules.

Xurmo generates
Spark SQL
statements.



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, database technology, and data virtualization. He works for R20/Consultancy (www.r20.nl), a consultancy company he founded in 1987.

Rick is chairman of the annual European Enterprise Data and Business Intelligence Conference (organized annually in London).

He writes for SearchBusinessAnalytics.Techtarget.com, B-eye-Network.com¹ and other websites. He introduced the business intelligence architecture called the *Data Delivery Platform* in 2009 in a number of articles² all published at BeyeNetwork.com. The Data Delivery Platform is an architecture based on data virtualization.

He has written several books on SQL. Published in 1987, his popular *Introduction to SQL*³ was the first English book on the market devoted entirely to SQL. After more than twenty five years, this book is still being sold, and has been translated in several languages, including Chinese, German, and Italian. His latest book⁴ *Data Virtualization for Business Intelligence Systems* was published in 2012.

For more information please visit www.r20.nl, or email to rick@r20.nl. You can also get in touch with him via LinkedIn and via Twitter [@Rick_vanderlans](https://twitter.com/Rick_vanderlans).

¹ See <http://www.b-eye-network.com/channels/5087/articles/>

² See <http://www.b-eye-network.com/channels/5087/view/12495>

³ R.F. van der Lans, *Introduction to SQL; Mastering the Relational Database Language*, Fourth Edition, Addison-Wesley, 2007.

⁴ R.F. van der Lans, *Data Virtualization for Business Intelligence Systems*, Morgan Kaufmann Publishers, 2012.



About Xurmo Technologies

Xurmo Technologies is a data infrastructure and analytics company, headquartered in Bangalore, India. The company was founded in 2009 by Sridhar Gopalakrishnan in close collaboration with scientists at Carnegie Mellon University, Pittsburgh.

Xurmo is also the name of their Hadoop-based DBMS. The Xurmo platform was launched in early 2014 after five years of R&D in the areas of data modeling, data architecture, machine learning, and personalization. The company has fifteen patent applications under prosecution at the USPTO and two patents have been awarded.

Xurmo has been deployed in production at various top tier companies in India including one of India's largest private banks; one of India's top three consumer goods conglomerate and a tier-1 consulting and knowledge process outsourcing company. Xurmo entered the US market in 2014 and has been deployed at Fishbowl Inc., a leading CRM and analytics solutions vendor in the hospitality industry.



Xurmo Technologies | Core Infrastructure for Big Data

www.xurmo.ai

