Data Replication for Enabling Operational BI

Whitepaper on Business Value and Architecture

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

June, 2012

Sponsored by

INFORMATICA
The Data Integration Company™

Table of Contents

1 Summary

Operational business intelligence (BI) is about supporting decision-making processes that require access to zero-latency data. Because these decisions are time-sensitive and often urgent, access to 'old' data is worthless. Most current business intelligence systems do not and cannot support operational BI. This whitepaper describes the importance of operational BI for modern-day organizations and explains how data replication *is a vital technology for enabling operational BI. In other words, the business value of deploying replication is to let business users operate with and to let them make decisions on zero-latency, operational data. This technology makes it possible for BI to follow the speed of today's businesses. Especially extended with* data virtualization*, data replication delivers some of the key building blocks required to develop operational BI systems.*

Every business intelligence specialist has noticed it: the world of BI is changing. On one hand there is a technology push. New technologies, such as data warehouse appliances, self-service BI tools, and NoSQL database servers, have become available, and, as always with new technology, it creates new opportunities; just remember the introduction of the steam engine and the airplane. Today, we can develop BI systems that were unthinkable a few years ago. On the other hand, the business pull is altering BI as well. New forms of reporting and analytics are demanded by the fast moving world of decision-making. This whitepaper focuses on one of these forms: *operational business intelligence*.

Operational BI implies supplying business users with operational data.

In a nutshell, operational BI involves reporting and analytical applications that support decision-making processes with operational data (sometimes referred to as *real-time data* or *zero-latency data*). More and more user groups can benefit from having access to operational data. Operational management, the workforce, and external business parties, such as customers and suppliers, are examples of such groups. Organizations usually identify the following reasons for implementing operational BI: improved efficiency of individual business processes and the organization as a whole, reduction of operating costs, and increased customer satisfaction.

In many ways, operational reporting and analytics for these user groups are not much different from the popular BI forms we have known for so many years. They also want data to be presented as 3D graphs and dashboards, they also want to be able to drill down and roll up the data, and they also demand interactive features. But there is one thing in which their needs differ from what we are used to: they need access to operational data. In fact, for them being able to access data that is a few hours late, let alone a few days, is useless. They have to work with the latest data.

The business value of replication is that it allows business users to operate with zero-latency data.

This whitepaper describes various architectures for developing business intelligence systems that present operational data. It focuses on a key enabling technology called *data replication*. The positive and negative aspects of each of the potential architectures are described and for each the role of data replication is explained. In most of these architectures, data replication is used to deliver

operational data to the data store used for reporting and analytics. In other words, the *business value* of deploying replication is to let business users operate with and to let them make decisions on zero-latency, operational data.

In this whitepaper, *data virtualization* technology is also addressed as an additional technology for developing operational BI. Data virtualization supports on-demand data integration and data transformation. It's especially these features that compliment data replication well. Together they form an ideal platform for developing operational BI systems.

One section is devoted to Informatica's product for data replication which offers the following unique set of features: heterogeneous database deployment (including high-end analytical and big data platforms), high availability, full range of configurations (including many-to-many), and console-based management.

2 Operational Business Intelligence: The Business Value of Replication

Operational BI requires access to operational data.

Most current business intelligence architectures offer users access to data that is one day, one week, or maybe even one month old. For a long time this was adequate for most users. Nowadays, more and more users are demanding access to data that is (almost) 100% up-to-date. In other words, they want to work with operational data. This new form of reporting and analytics is usually referred to as o*perational reporting and analytics*, or *operational BI* for short.

There are many examples of environments that need operational reporting and analytics. For example, a retail company may want to know whether a truck, already on the road to deliver goods to a specific store, should be redirected to another store that has a sudden, more urgent need for those products. It would not make sense to execute this analysis with yesterday's data. Another example is credit card fraud detection. A classic form of credit card fraud is when stolen card data is used to purchase products. Each new purchase has to be analyzed to see if it fits the buying pattern of the card owner and whether the purchase makes sense. One of the checks could be whether two purchases in different cities occurred within a limited time of each other. For example, if a new purchase is made in Boston and the previous one was in San Francisco just a few seconds earlier, it's very likely this is a case of fraud. But this form of analysis only makes sense on operational data and additionally requires some way to resolve data inaccuracies and inconsistencies instantaneously. And for almost every organization, commercial or non-commercial, examples can be given where operational BI can improve the business operations.

The business value of replication is that it allows business users to operate with zero-latency data.

To summarize, the *business value* of deploying replication is to let business users operate with and to let them make decisions on zero-latency, operational data. Replication makes it possible for BI to follow the speed of today's businesses. The business will not be constrained anymore because only yesterday's data can be obtained. They can see what's happening in the business processes live.

3 Different Forms of Operational Business Intelligence

To illustrate the wide range of application areas of operational BI, four forms of operational BI are described here:

Operational Reporting – This form of operational BI presents live data to show the status and progress of particular business processes. This data is usually presented in a dashboard so that managers can see at a glance what the status is and whether actions should be taken.
An example of such an application is a factory where a dashboard shows the percentage of spill in a factory. If the percentage becomes too high, immediate action has to be taken to reduce the spill to minimize unnecessary costs. For operational reporting, access to the most up-to-date data is crucial, because delays could harm an organization.

Operational Analytics – With operational analytics, operational data is merged with historical data to show whether what's happening is in line with historical patterns. In other words, is the organization doing better or worse than in the past? Operational analytics can also be used to predict what can be done.
Returning to the factory example, showing what's happening is quite useful, but it's even more helpful if the dashboard also shows what is normal, and what's not. Showing that spill is up to 4% becomes more meaningful if the average spill percentage of, for example, 2% is also presented.

Embedded Analytics – With embedded analytics, complex analytical and statistical models are developed that run embedded within production systems. An example could be a call center application used by an insurance company. This application allows operators to retrieve data on the calling customers. Such an application can be extended with information showing whether it would make sense to (cross-)sell this customer other services. In an ideal solution, the application shows which cross-sell opportunities exist and what the chance is that this customer may be interested. So, instead of selling the customer what the insurance company wants, they try to sell what may be of interest to the customer. Predictive models can be developed for making those cross-sell predictions. When finished they are embedded inside the call-center application. For these models to run correctly, they need access to operational and historical data.

With embedded analytics, predictive models run within production systems and access operational and historical data.

Big Data Analytics – A relatively new form of analytics is called *big data analytics*. Many traditional information systems exist today that store and manage large numbers of records. More recently, new systems have been built that store an amount of data magnitudes larger than those in traditional systems. For example, click-stream applications, sensor-based applications, and voice and image processing applications, they all generate massive numbers of records per day. The amount of records here is not measured in millions, but sometimes in trillions. Analyzing this amount of data is a challenge.
An example of an organization that is processing massive amounts of data is a European utility company currently installing 55 million meters in as many houses that measure and report energy consumption every 15 minutes. This means that every 15 minutes 55 million measurements are processed and stored. This amounts to over 5 million records per day and to

close to 2 trillion records per year. This amount of data can clearly be classified as *big data*. More and more organizations are developing production systems with comparable amounts of data. In most of these big data environments, the need also exists to analyze all that data when it's entered. So, big data analytics involves analyzing massive amounts of operational data. Therefore, we classify it as a form of operational BI.

4 Are Classic Business Intelligence Systems Ready for Operational BI?

In most business intelligence systems, data in the data warehouses and/or data marts is refreshed periodically: every month, week, day, or so many hours. This means that the data available to the users is not operational data. When new data is entered in the production systems, it may take some time before that data arrives in the data warehouse and becomes available for reporting. This makes most business intelligence systems unsuitable for operational reporting and analytics.

The problem is related to the way source data is copied and integrated. The two most common technologies for data integration and transformation software used in business intelligence systems are *ETL* (Extract Transform Load) and *ELT* (Extract Load Transform). The essential difference between these two is the order in which operations take place. With ETL, data is first extracted from a source, next the data is transformed, and finally the result is stored in a target database. With ELT, the order of the last two operations is switched. Extracted data is first stored, next it's transformed, and finally, the transformed data is stored (again).

For many forms of reporting and analytics ETL and ELT are the right solutions.

In numerous business intelligence systems, ETL or ELT solutions are used to copy data from production systems to a data staging area or to a data warehouse, or from a data warehouse to a data mart. The main characteristics of both are: scheduled data copying, batch-oriented processing, simple to advanced data cleansing, transformation, and integration operations. These products are developed and optimized to copy data from a source to a target system periodically. For many forms of reporting and analytics and also for many users, ETL and ELT are the right solutions. These technologies have proven their worth over and over again in many business intelligence systems. However, it's difficult to develop operational reporting applications when data is copied periodically and no access to operational data is possible.

Besides ETL and ELT other forms of data integration exist, such as data replication and data virtualization. These forms enable operational BI. The next section describes data replication and Section 6 explains data virtualization.

5 What is Data Replication?

As with ETL and ELT, *data replication* (or replication for short) is also used for copying data from one data store to another. The difference with ETL and ELT is that instead of data being copied in a scheduled fashion, source data is copied the moment it's updated, inserted, or deleted; see

Figure 1. In other words, the moment data in the source system is changed, it's replicated to a target data store. Because of this instantaneous copying, the data in the target data store is in line with the data in the source database. It's a *replica*. This means that if the source data store contains operational data, so does the target data store.

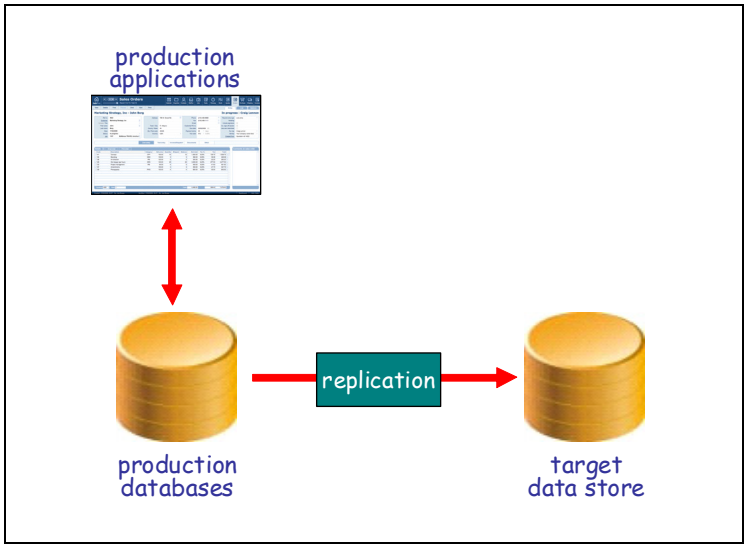

Figure 1 *Data replication can be used for instantaneous copying of new data from one data store to another*

Data Replication Keywords – The keywords that sum up data replication are: instantaneous, fast, event-driven simple transformations, no data integration:

Instantaneous: Because source data is replicated right after it has been changed and committed, there is barely any delay. The result is that a target data store can be close to 100% up-to-date. So, the target data store can be regarded as an operational database as well.

Data replication copies data right after it has been changed.

Fast: To shorten the delay and to keep up with the number of transactions on the source system, it's important that data replication products extract data fast, transform data fast, and load data in the target data store fast. When extracting data, these products should minimize potential interference on the source system, they should minimize concurrency problems, and they should not influence the performance of the applications accessing the source database.

Event-Driven: A replication process is not scheduled, but is event-driven. When, because of an event, data in the source system changes, a comparable change is applied to the target data store.

Simple Transformations: Although data replication is usually considered to be a form of data transformation as well (like ETL and ELT), its transformation features are limited. The products can apply simple filters, suppress certain columns, and transform column values, but no sophisticated data transformation and cleansing operations are supported. The main reason is that they have to copy fast. Too many and too complex transformations can slow down the replication process considerably. In a way, data replication can be classified as EtL (with a

Data replication can be classified as EtL (small t).

small t), meaning data replication can extract data fast, it can load fast, but it supports limited transformation features.

No Data Integration: The term data integration usually refers to bringing data together from multiple source systems. Data replication tools support data integration as well, but here the same applies as for data transformations: data integration slows down the replication process too much. Data from data sources is usually integrated after replication. In this case, replication forms the E and the L of ELT.

Change Data Capture – If a data replicator accesses a database itself, it requests the database server to retrieve data. This has as effect that the database files are accessed. This may cause unwanted interference on the source system.

A less interfering approach is if the data replicator reads the log files of the database server. This is usually referred to as *change data capture* (CDC). Database servers write all the data changes made to log files. These files are used in case anything goes down, such as a crash of the server itself, in order to recover and to bring the database back in a correct state.

When change data capture is used, log files are read to minimize interference.

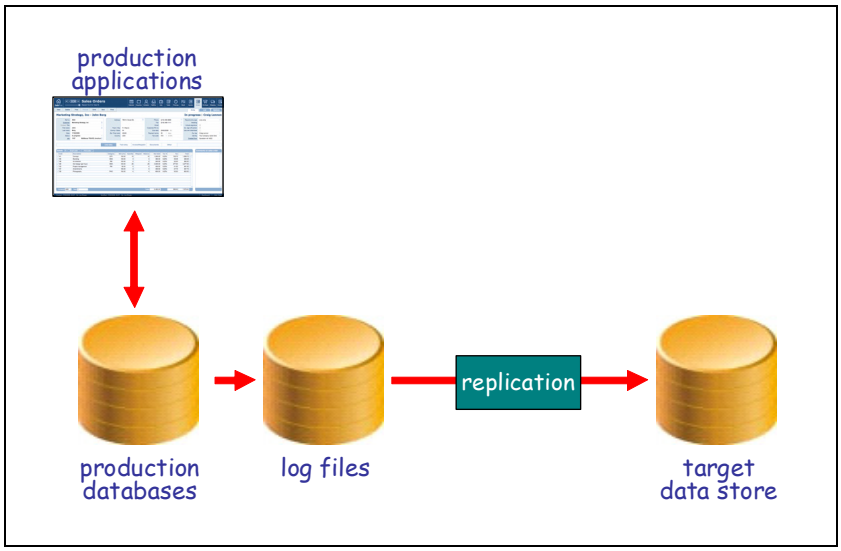

Figure 2 *Data replication using log files as input*

By reading log files there is really no interference with the real production work. The data replicator constantly inspects the log files and when relevant data is committed, it's copied from the log files to the target data store. Technologically, this is quite complex, but it guarantees a low level of interference.

Iceberg Technology – Data replication can be qualified as *iceberg technology*. To make the tools work properly, developers of data replication tools have to understand the structure of log files, how transactions are being committed, and so on. They have to understand the internal workings of the database servers up to the smallest detail. The developers using the replicators tools don't.

Data replication is iceberg technology; much of the complexities are hidden for the developers.

All this complexity is hidden for them. Developers see a simple user interface where they can define which data to replicate and how the replication should take place. Most of the advanced technology stays hidden underneath the hood. Developers never see how replicators read the log files and how they manage to identify committed transactions. Conclusion, developers only see a very small portion of these products, in the same way only the tip of an iceberg is visible above the waterline. Data replicators are complex and advanced on the inside, but simple on the outside.

Data replication as a technology has existed for many years. Until recently it was primarily used in the world of production systems for application areas, such as:

- Increasing availability and disaster recovery
- Distribution of workload
- Operational data synchronization

But with the growing interest in operational BI, replication is more and more seen as an enabling technology for developing operational BI systems. In Section 7 this topic is discussed in detail.

6 What is Data Virtualization?

Data virtualization is the fourth form for integrating and transforming data described in this whitepaper. In a nutshell, when data virtualization is applied, an abstraction and encapsulation layer is provided that, for applications, hides most of the technical aspects of how and where data is stored; see Figure 3. Because of that layer, applications don't need to know where all the data is physically stored, how the data should be integrated, where the database servers run, what the real table structures are, what the required APIs are, which database language to use, and so on. When data virtualization technology is deployed, to every application it feels as if one large database is accessed. Data virtualization offers a unified view of all the data.

Data virtualization makes a heterogeneous set of data stores look like one integrated data store.

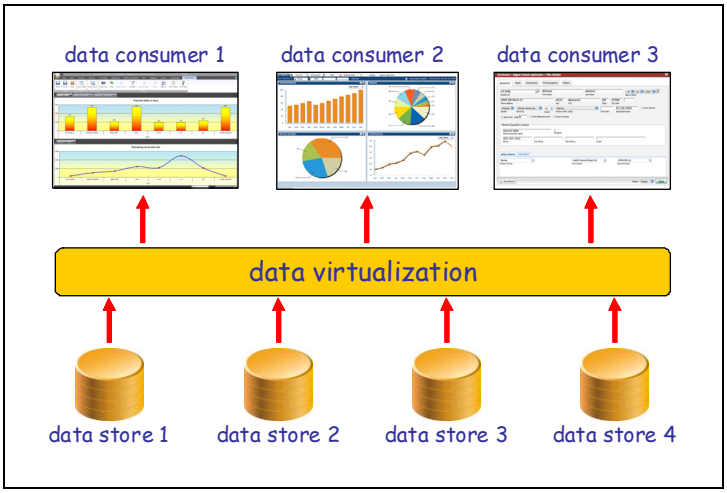

Figure 3 *When data virtualization is applied, all the data stores are presented as one integrated data store (unified view)*

In Figure 3 the terms *data consumer* and *data store* are used. The neutral term data consumer refers to any application that retrieves, enters, or manipulates data. For example, a data consumer can be an online data entry application, a reporting application, a statistical model, an internet application, a batch application, or an RFID sensor. Likewise, the term data store is used to refer to any source of data. This data source can be anything, such as a table in a SQL database, a simple text file, an XML document, a spreadsheet, a web service, an index sequential file, and an HTML page.

The definition of data virtualization:

> *Data virtualization is the technology that offers data consumers a unified, abstracted, and encapsulated view for querying and manipulating data stored in a heterogeneous set of data stores.*

So, even if the data stores in Figure 3 use different storage formats and technologies, to the data consumers, the data virtualization layer presents them all as one integrated set of data. In addition, different languages can be used to access the same data.

When ETL, ELT, and data replication are used, the result of data integration, cleansing, and transformation is stored before it can be used for reporting. Data virtualization, on the other hand, integrates, cleanses, and transforms data *on-demand*. In other words, when the data is retrieved, only then data is processed. The result is not stored, but passed to the reporting application.

Table 1 shows the differences between data replication and data virtualization. What's obvious from this table is that these two forms are very complimentary. For example, data replication has the ability to create a copy of the source data fast, while data virtualization is strong in data integration and transformation.

Data Replication	Data Virtualization
Requires target data store or file for storing result	No extra data stores required, result is passed to data consumer
Fast and efficient access of data source	Repeated access of data source
Access of log files to minimize interference	Access of source data which can cause interference
Limited data transformation features	Extensive data transformation features
Limited data integration features	Extensive data integration features
Event-driven data replication	Query-driven data transformation and integration

Table 1 *High-level comparison of data replication and data virtualization*

Data virtualization has evolved from *data federation technology*, which was introduced in the 1990s. While data federation was primarily runtime technology for applications to access multiple databases as one logical database, data virtualization supports many more features. Evidently, most data virtualization products support more data store technologies than their forerunners, data federation and they have been optimized to handle the query and data workloads needed today. But with respect to supporting the entire development lifecycle, *advanced data virtualization* products have much more to offer, including the following features (see also Figure 4):

- Designers can define *canonical data models* in data virtualization servers. In other words, data virtualization supports top down development, the virtual table structures can be designed the way the applications want to see them.

- Data virtualization servers support *on-demand* (or real-time) *data profiling*. When a virtual table has been defined, it's virtual contents can be profiled by simply pushing the button.

- Data virtualization tools integrate and transform data live; they allow simple and complex data cleansing operations to be invoked live as well (if demanded by the designers).

- Because designers can develop canonical data models and invoke on-demand data profiling, data virtualization servers support *collaborative* (designer and business users together) and *iterative* styles of development. The specifications entered in data virtualization servers are easy to change; it's a forgiving technology.

- Self-service BI tools, such as Qlikview, Spotfire, and Tableau, have become very popular. They allow business users to develop and change their existing reports themselves. Still, their level of self-serviceness is restricted, because users are not allowed to change existing data mart or data warehouse structures; this is the responsibility of the IT department. A BI system with a data virtualization server at the heart, is more flexible; it's like promoting self-service BI to *managed self-service BI*.

- Data virtualization servers have the right features to support *enterprise-wide deployment*, such as security, high-performance, scalability, and robustness,

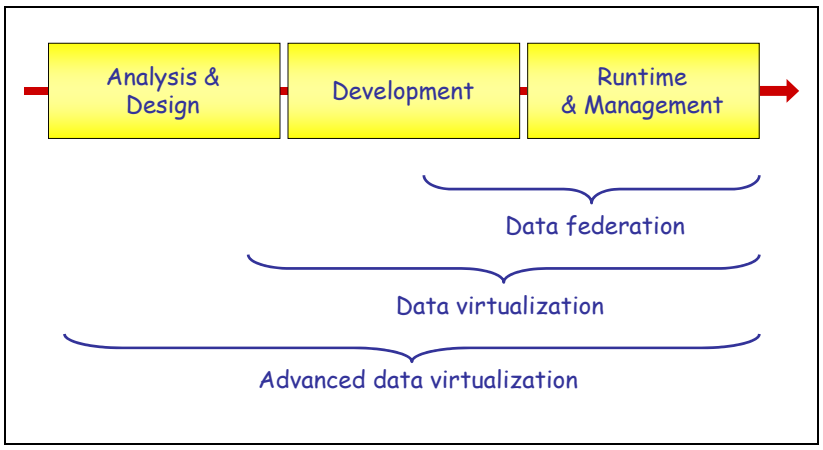

Figure 4 *The evolution from data federation to advanced data virtualization*

The general expectation is that advanced data virtualization products will continue to evolve and to support many more aspects of the system development life cycle, including business glossaries, master data management, and versioning of data models.

For more information on data virtualization, see the whitepaper *Developing a Data Delivery Platform with Informatica Data Services*[1]. In addition, see also the upcoming book *Data Virtualization for Business Intelligence Systems*, which will be published in the summer of 2012 by Morgan Kaufmann Publishers.

7 Four Architectures for Operational BI

Operational reporting and analytics need access to operational data. Operational data is entered in the production systems, so somehow this data has to be made available to the users. This section describes four architectures that support operational BI. Except for the first architecture, data replication is used in most of these architectures. In fact, in the other three architectures data replication forms the booster of operational BI. For each architecture the strong and weak points are identified and at the end of the section a summary is shown.

Architecture 1 – Operational BI on Production Databases

Architecturally, the simplest way of developing operational reports that need access to operational data, is by building them straight on the production databases; see Figure 5. Because the reporting applications are accessing the latest production data, they can show what's happening at that particular moment, they can present the status and progress of various business processes.

This architecture has the following positive characteristics.

- Access to Operational Data: The reporting applications access production databases and can therefore show operational data to the users.

- Simple Architecture: This architecture consists primarily of reporting applications and possibly middleware technology to connect the reporting tools to the production databases.

However, this architecture, although very simple, has some negative characteristics as well:

- High Interference: When reporting applications run queries to retrieve data, they interfere with the processing of the production applications. Both applications will compete for shared resources. The users of the production applications may experience a performance decrease. The amount of decrease depends on the transaction and query workload.

[1] R.F. van der Lans, Developing a Data Delivery Platform with Informatica Data Services, 2011, http://www.informatica-urg.com/rick-f-van-der-lans-white-paper-developing-a-data-delivery-platform-with-informatica-data-services-a-technical-white-paper-on-next-generation-data-virtualization/

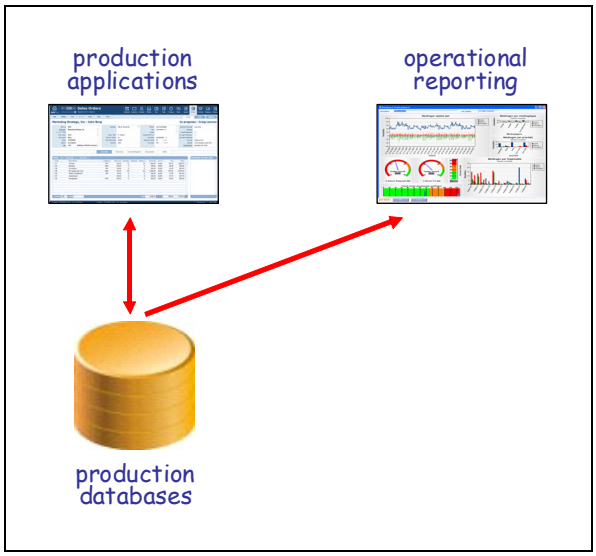

Figure 5 *Operational BI without data replication; the reports are accessing the production databases directly*

- **Low Concurrency:** Depending on the database server in use, queries coming from the reporting applications and transactions coming from the production applications may cause dramatic data locking problems causing concurrency issues. This can lead to situations where users have to wait for locks to be freed before they can continue.

- **Poor Query Performance:** The queries of reporting applications can be complex and can take quite a long time to process, especially if the database server also has to process the transaction workload concurrently. In addition, production databases are tuned and optimized for running transactions and not for running reporting queries, therefore they can't offer the best query performance.

- **Coded Data:** To keep production databases as small and efficient as possible, a lot of their data is heavily coded. In addition, the meaning of those codes is not always stored in the database, and, in some cases, the meaning is even hidden in the application code. This makes reporting quite complex. Logic has to be added to the reports to transform the codes in data that is meaningful for the users. Not all reporting tools support the right features to implement such transformations.

Reporting tools usually have no or limited features for decoding data, correcting data and for transforming data.

- **Incorrect and Inconsistent Data:** The quality of data stored in production databases is not always perfect. Some data values are incorrect or inconsistent. This is why such a large portion of a data warehouse budget is usually spent on cleansing data. In this first architecture, the reporting applications access incorrect data and, therefore, they are responsible for cleansing the data themselves. Unfortunately, most reporting tools have no or limited capabilities for this type of operation.

- **Complex Database Schema:** The structures of the tables and values in the columns of production databases have been designed to comply with the needs of the production applications. Data is probably stored in tables with highly normalized structures.

Unfortunately, those structures are not suitable for the reporting applications. Again, code has to be added to the reporting applications to deal with these unfitting table structures. This is hard to implement in most tools. Some reporting tools even demand that data is organized as star schemas.

- **Complex Data Integration:** As indicated, data replicators are not good at integrating data, which means the reporting applications have to do the integration work themselves. More and more reporting tools support features for data integration, but certainly not all.

- **Specification Proliferation:** Each reporting application accessing a replicated database has to include specifications to deal with the previously mentioned issues, such as incorrect data, complex database schema, and complex data integration. This can lead to replication of specifications across reporting tools and applications. This proliferation of specifications is hard to manage and can lead to inconsistent reporting results.

- **Missing Historical Data:** If we assume that historical data is not kept in the operational database, then combining operational data with historical data for, for example, trend analysis, is not possible with this architecture.

Note: In this section we assume that classic database servers are used by the production applications. New database servers, such as MongoDB and VoltDB, offer new transaction mechanisms that have fewer problems with running queries and transactions concurrently.

Architecture 2 – Operational BI using Data Replication

In the second architecture, the reporting applications access replicated production databases and not original production databases; see Figure 6. In this architecture, we assume that log files are read and not database files. Depending on how the replication technology is set up, the reporting applications access operational data. If there is a short delay with respect to replication and if that delay is acceptable for the reporting applications, these applications can show the live status and progress of various business processes.

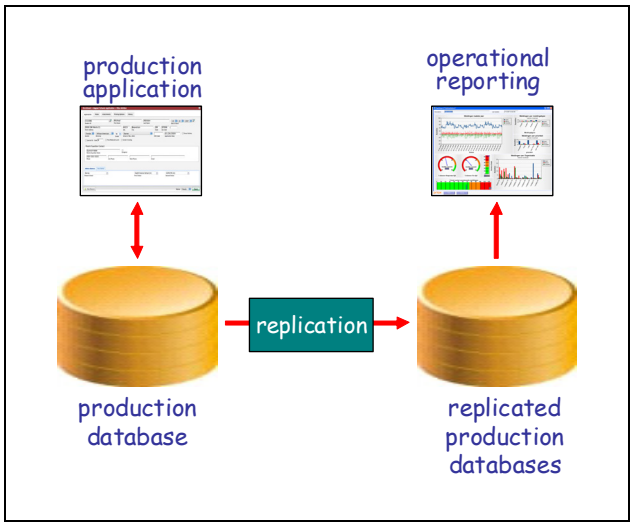

Figure 6 *Operational BI using data replication*

This architecture has the following positive characteristics.

- **Access to Operational Data:** The reporting applications have access to operational data (or data almost 100% up-to-date) and can therefore show operational data.

- **Medium-complex Architecture:** The architecture consists primarily of reporting applications, replication technology, and possibly middleware technology to connect the reporting applications to the replicated databases.

- **Low Interference:** When reporting applications run queries to retrieve data, they don't interfere with the production applications. Users of the production applications don't experience a performance decrease, because the two user groups access different databases.

- **High Concurrency:** Depending on the database server in use, queries coming from the reporting applications and transactions coming from the production applications don't cause data locking problems, therefore causing no concurrency issues. The only problem is that the data replicator, feeding data into a replicated database, may interfere with the queries, but this has no impact on the production systems.

- **Good Query Performance:** Although the structures of the tables in a replicated database are probably very identical to those of the source database, the former can be optimized and tuned very differently: different indexes can be defined; table space parameters can be set differently, the buffer can be optimized differently, and so on. Because it's a separate database server, it can be tuned towards query processing.

The structure and the contents of the source and the target database are very similar (limited or no transformations take place when replicating data), therefore most of the negative characteristics of the first architecture apply to the second as well:

- **Coded Data:** Code has to be added to the reporting applications to transform the codes in data values that are meaningful for users. Not all reporting tools support the right features to implement those transformations.

- **Incorrect and Inconsistent Data:** The reporting applications are responsible for cleansing the data. Unfortunately, most reporting tools have no to limited capabilities for data cleansing.

- **Complex Database Schema:** Code has to be added to the reporting applications to deal with the non-fitting table structures. This is hard to implement with most reporting tools.

- **Complex Data Integration:** The reporting applications are responsible for integrating data from multiple data sources. More and more reporting tools support features for data integration, but certainly not all.

In architecture 2 the reporting applications are responsible for decoding data, cleansing data, data integration, and schema transformation.

- **Specification Proliferation:** As with the first architecture, each reporting application accessing a replicated database has to include specifications to deal with the issues above. This can lead to replication of specifications across tools and applications. This proliferation of specifications is hard to manage and can lead to inconsistent reporting results.

- **Missing Historical Data:** The target data store is a replica, thus, if the source doesn't contain historical data neither will the target data store. This makes it impossible to develop reports that need historical data.

This second architecture is certainly an improvement over the first one. Most of the problems dealing with performance, scalability, and concurrency are solved. However, issues related to the data structures and data remain.

Architecture 3 – Operational BI using Data Replication and Data Virtualization

The difference between this third architecture and the previous one is that a data virtualization server is used by the reporting applications to access the replicated databases; see Figure 7. A data virtualization server supports the features to transform the data structure of the replicated database to one that fits the needs of the reporting applications. In other words, in this architecture the data virtualization server is responsible for most of the data integration, data transformation, and data cleansing tasks.

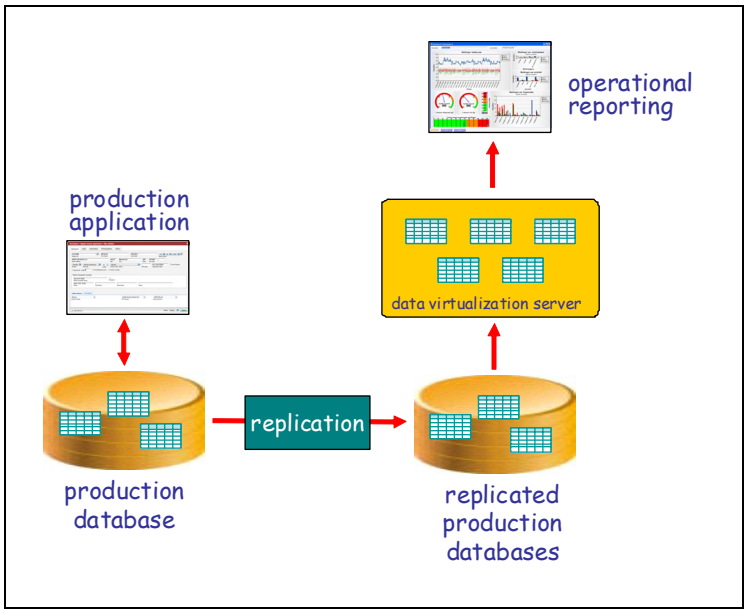

Figure 7 *Operational BI using data replication and data virtualization*

In this third architecture, because the data virtualization server retrieves data from the replicated databases in an on-demand fashion, reporting applications see operational data. So data is retrieved from the replicated production databases when the applications ask for it, and not sooner. The result coming from the data virtualization server is therefore as up-to-date as the data stored in the replicated databases.

This architecture has the following positive characteristics.

- **Access to Operational Data:** The reporting applications have access to operational data (or data almost 100% up-to-date) and can therefore show operational data.

- **Medium-Complex Architecture:** The architecture consists primarily of reporting applications, replication technology, and data virtualization technology to connect the reporting applications to the replicated databases.

- **Low Interference:** When the reporting applications run queries to retrieve new data, they don't interfere with the production applications. Users of the production applications will not experience a performance decrease, because the two user groups are accessing different databases.

- **High Concurrency:** Depending on the database server in use, queries coming from the reporting applications (via the data virtualization server) and transactions coming from the production applications will not cause data locking issues, therefore causing no concurrency issues. The only problem is that the replicator feeding data into the replicated databases, may interfere with the queries, but this will have no impact on the production systems.

- **Good Query Performance:** Although the structures of the tables in the replicated databases are probably very identical to those of the source databases, the former can be optimized very differently. Different indexes can be defined, table space parameters can be set differently, the database buffer can be optimized differently, and so on. Because it's a separate database server, it can be tuned for query processing.

- **Hiding of Coded Data:** A data virtualization server is capable of transforming coded data to more meaningful data values. This is transparent to the reporting applications.

- **Hiding of Incorrect and Inconsistent Data:** A data virtualization supports advanced features for cleansing data. This is transparent to the reporting applications.

- **Hiding of Complex Database Schema:** A data virtualization supports features to transform the table structures. For example, a set of normalized tables can be transformed into a star schema. This transformation is transparent to the reporting applications.

- **Hiding of Complex Data Integration:** A data virtualization supports features for integrating data from multiple databases, including from a heterogeneous set of databases.

- **Specification Centralization:** With data virtualization all the specifications for data integration, data transformation, data decoding, and data cleansing are stored centrally, and are shared by all the reporting applications, hereby minimizing the proliferation of specifications.

This architecture has one negative aspect:

- **Missing Historical Data:** The target data store is a replica, thus if the source database doesn't contain historical data neither will the target data store. This makes it impossible to develop reports that need historical data as well.

By adding data virtualization to the solution, the missing features for, for example, data transformation and data integration are added. Due to their complimentary features, data replication and data virtualization form a perfect combination for delivering operational BI.

Note: An alternative architecture to architecture 3 is one where data replication is not used, but reporting applications access the production databases via a data virtualization server. All the above positive characteristics apply to this alternative architecture as well. However, because the production databases are accessed live, there will be problems with interference, concurrency, and performance.

Architecture 4 – Operational BI using Replication, Data Virtualization, and a Data Warehouse

The three previous architectures all allow the reporting applications to process operational data. Some reports need historical data as well. By extending the architecture with a data warehouse, this historical data becomes available; see Figure 8. In this architecture, a data virtualization server is responsible for integrating the production and historical data and for presenting the reporting applications with one unified view of all the data.

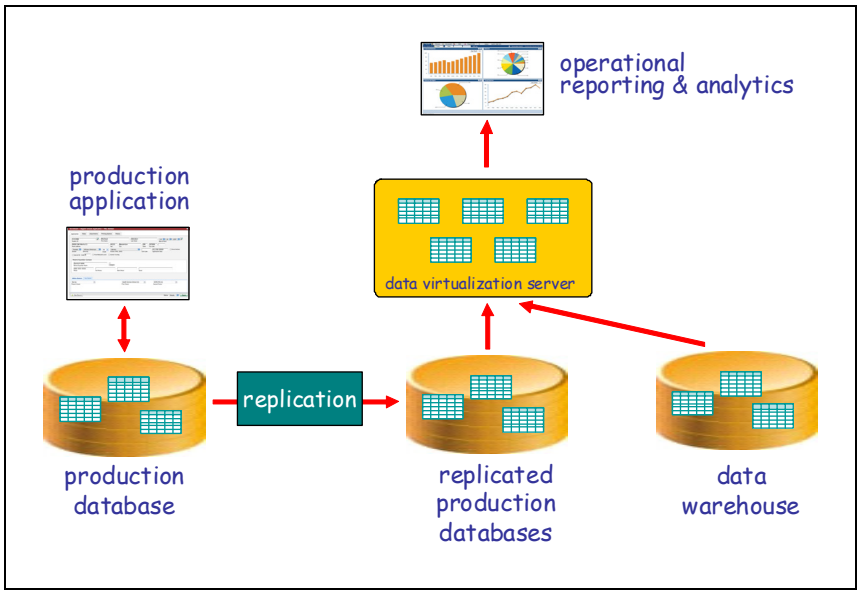

Figure 8 *Operational BI using replication, data virtualization, and a data warehouse*

This architecture has the following positive characteristics.

- **Access to Operational Data:** The reporting applications have access to operational data (or data almost 100% up-to-date) and can therefore show operational data.

- **Complex Architecture:** The architecture consists of reporting applications, replication technology, and data virtualization technology to connect the reporting applications to the replicated databases and a data warehouse.

- **Low Interference:** When the reporting applications run queries to retrieve new data, they don't interfere with the production applications. Users of the production applications will not experience a performance decrease, because the two user groups are accessing different databases.

- **High Concurrency:** Depending on the database server in use, queries coming from the reporting applications (via the data virtualization server) and transactions coming from the production applications will not cause data locking issues, therefore causing no concurrency issues. The only problem is that the replicator feeding data into the replicated databases may interfere with the queries, but this will have no impact on the production systems.

- **Good Query Performance:** Although the structures of the tables in the replicated databases are probably very identical to those of the source databases, the former can be optimized very differently: different indexes can be defined; table space parameters can be set differently, the buffer can be optimized differently and so on. Because it's a separate database server, it can be tuned towards query processing.

- **Hiding of Coded Data:** A data virtualization server is capable of transforming coded data to more meaningful data values. This is transparent to the reporting applications.

- **Hiding of Incorrect and Inconsistent Data:** A data virtualization supports advanced features for cleansing data. This is transparent to the reporting applications.

- **Hiding of Complex Database Schema:** A data virtualization supports features for transforming the table structures. For example, a set of normalized tables can be transformed into a star schema. This is transparent to the reporting applications.

- **Hiding of Complex Data Integration:** A data virtualization supports features for integrating data from multiple databases, including from a heterogeneous set of databases. This is transparent to the reporting applications.

- **Specification Centralization:** With data virtualization all the specifications for data integration, data transformation, data decoding, and data cleansing are stored centrally, and are shared by all the reporting applications, hereby minimizing the proliferation of specifications.

- **Availability of Historical Data:** The data warehouse contains all the required historical data, thus making it possible to develop of report applications that need historical data. An important requirement is that key values in the production databases and the data warehouse conform.

By adding access to the historical data in a data warehouse, the architecture can support any form of operational BI, including the one that needs to integrate operational and historical data. But, as with the previous architecture, the data replication and data virtualization technologies form the core building blocks, the glue that links it all together.

Summary Alternative Architectures for Operational BI

To summarize the four architectures described in the previous sections, Table 2 contains an overview.

	Architecture 1 – Without Replication	Architecture 2 – With Replication	Architecture 3 – With Replication and Data Virtualization	Architecture 4 – With Replication, Data Virtualization, and Data Warehouse
Access to operational data	Yes	Yes	Yes	Yes
Architecture	Simple	Simple	Medium	Complex
Decreased interference		Yes	Yes	Yes
Improved concurrency		Yes	Yes	Yes
Improved query performance		Yes	Yes	Yes
Hiding coded data			Yes	Yes
Hiding incorrect and inconsistent data			Yes	Yes
Hiding complexity of database schema			Yes	Yes
Hiding data integration			Yes	Yes
Centralizing specifications			Yes	Yes
Supply of historical data				Yes

Table 2 *Comparison of four architectures supporting operational BI*

8 Integrating Big Data with Data Replication and Operational BI

Big Data Stores and NoSQL – Operational BI is not the only new form of BI that's changing the BI landscape. Another popular form, with the intriguing name *big data analytics,* has at least a comparable impact. As the term suggests, big data analytics means analyzing massive amounts of data; see also Section 2.

Big data analytics means analyzing massive amounts of data.

Due to the sheer size of big data stores, analyzing big data can be a technological challenge. In some cases, the amount of data is too much to handle for classic database servers. Therefore, a new class of data storage products has recently been introduced. Due to their non-relational data models, different storage and transaction mechanisms, and due to the way they parallelize processing, they are capable of supporting bigger amounts of data and therefore making big data analytics possible. In this whitepaper we refer to them as *big data stores*.

What binds most of these products is the fact that they don't support SQL or that SQL is not their primary database language. That's why the catchy name *NoSQL* is used to refer to them. In the beginning, NoSQL stood for *No SQL*, implying the data managed by these products was not accessible using SQL. They supported their own database language(s) and API(s). Nowadays, because more and more of these products offer some minimal form of SQL, the acronym NoSQL stands for *Not Only SQL*.

Examples of well-known big data stores are Apache Hadoop, Apache Cassandra, CouchDB, 10Gen MongoDB, and InfiniteGraph. Especially of Apache Hadoop many distributions exist, such as the ones from Cloudera, HortonWorks, IBM, MapR, and Oracle. Cassandra is also available from DataStax.

Differences with SQL Database Servers – Several differences exist between classic SQL database servers and big data stores. Here we list a few that relate to business intelligence:

- With respect to the data model, some big data stores support non-relational concepts such as *repeating groups* (a set of values within a column-row combination) and *hierarchies* (data values being part of other data values).
- Nearly all big data stores are efficient at storing and manipulating semi-structured data, such as URL strings, JSON and XML documents, and unstructured data, such as emails, tweets, and contracts.
- Most big data stores support *multi-schema tables*; sometimes referred to as poly-schema or poly-structure tables. Each row in such a table can have a different set of columns (and thus values).
- Due to the multi-schema concept new columns can be added to existing tables very easily, offering a high level of flexibility.
- To increase parallelization and scalability, most have implemented Google's *MapReduce*[2] algorithm. This improves load and query performance.

Using Big Data Stores – As indicated, in most cases, big data stores are used for storing and manipulating massive amounts of data. Production applications writing new data to a big data store usually have one of the following two architectures. In the first architecture, the production application writes directly to a big data store; see Figure 9. An example of such an application is a sensor-based application that gathers measurement data from a large set of sensors and that inserts all that data in a big data store. A reporting application queries the big data store directly to analyze the data.

In the second architecture, a production application writes to and reads from a classic database and in addition it logs every event in a separate big data store; see Figure 10. In this architecture as well, the reporting application queries the big data store.

[2] 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.

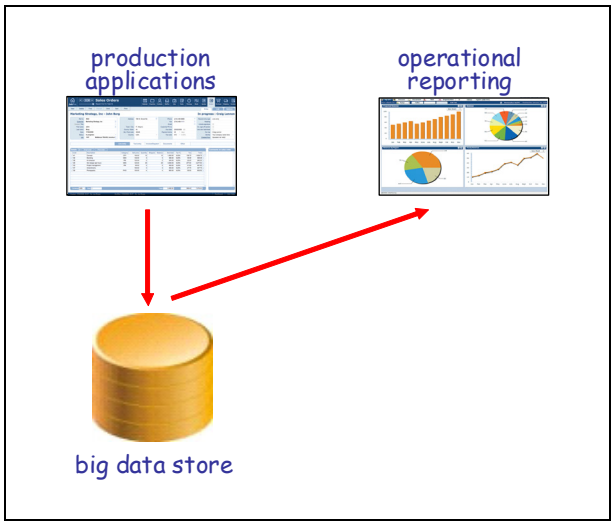

Figure 9 *A production application writing new data to a big data store*

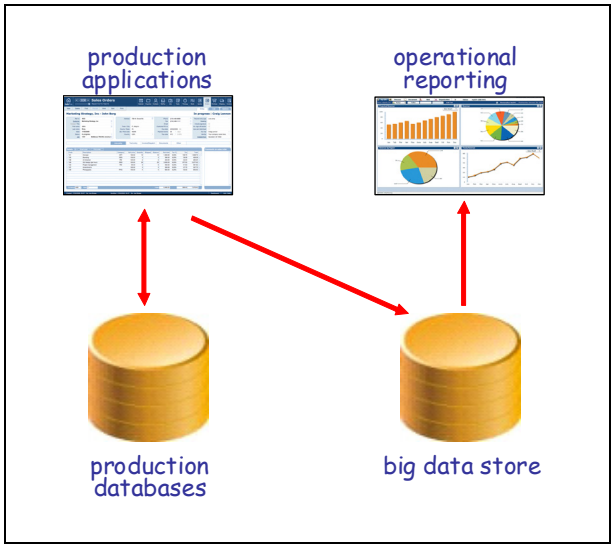

Figure 10 *A production application writing to production databases and logging data to a big data store as well*

Limitations of Big Data Stores – Both architectures have the same limitations:

- **Low-level Interface:** Most current big data stores offer a low-level programmatic interface. There is no or minimal support for SQL. Developing an analytic application that accesses a big data store requires more development time and requires more in-depth technical knowledge of the supported interface than when a SQL database server is used.

- **No Support for Popular BI Tools:** Most tools used by organizations for reporting and analytics expect a data source to support SQL. Therefore, most of them are not capable of accessing big data stores. The effect is that business users can't use their favorite tools.

- **Non-Relational Concepts:** The concepts mentioned in this section, such as hierarchies, semi-structured data, and multi-schema, can't be handled by most reporting tools. This means that when data is stored with those concepts, they can't be accessed by those

tools. First, that data has to be somehow transformed somehow into a relational format that leads to additional storage.

- **Low-Level Data:** To speed up inserting data in a big data store and to minimize the amount of data stored, data is usually coded. The consequence is that little or no descriptive data is stored in big data stores. Using the terminology of data warehousing, it's almost as if only fact data is stored and no dimensional data. If a reporting tool only has access to fact data in a big data store, it's impossible to transform the coded data to meaningful data for the business users. This seriously restricts reporting and analytics.

Most big data stores only contain fact data and not descriptive data restricting reporting and analytics.

The rest of the section describes how replication can be used to overcome the foregoing limitations.

Using Data Replication to Load Data in Big Data Stores – Some of the limitations can be overcome by copying descriptive data to the big data store; see the architecture in Figure 11. This descriptive data may be coming from the production databases or a data warehouse. In this architecture, replication technology can be used to copy new descriptive data when it becomes available.

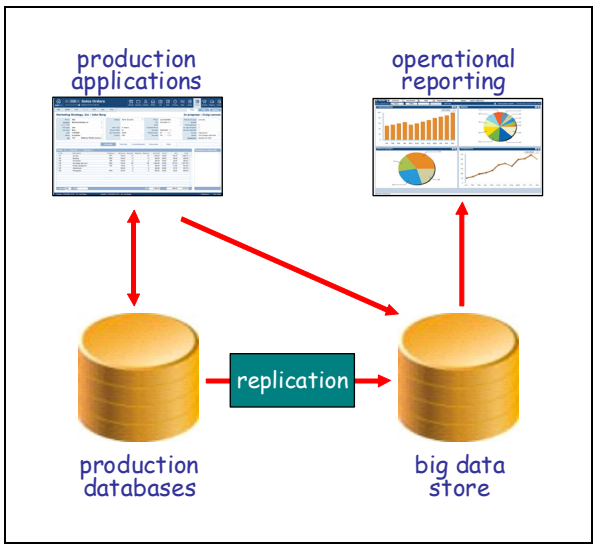

Figure 11 *Copying descriptive data from production databases to the big data store to turn coded data into more meaningful data*

As an example, one company is currently using data replication to extract terabytes of data per hour from its Oracle data warehouse and to load it into Hadoop. In Hadoop, they aggregate the data down to a lesser volume. After the volume has been reduced, it's loaded into a Teradata-based system, where it is used for reporting and analytics. Replicating it into Hadoop for aggregation is done to avoid the cost of upgrading their Teradata system to handle the massive volume of data that would result otherwise.

An alternative solution for getting the descriptive data in the big data store is to let the production applications insert new fact data and new descriptive data in the big data store. The

risk of this architecture is that the additional inserts of the descriptive data can seriously slow down the whole insert process, because inserting that descriptive data is part of the transaction. In the first solution, replication is used for adding descriptive data but this is done as a separate transaction.

In both architectures one problem is solved: because the coded data is extended with descriptive data it has become more meaningful. However, the other three problems remain: low-level interface, no support for popular BI tools, and use of non-relational concepts.

Using Replication to Extract Data from Big Data Stores – An architecture that doesn't have the four limitations is one where data is pushed the other way: from the big data store to the data warehouse environment; see Figure 12. This solution can be implemented when replication technology copies relevant data from the big data store to a staging area. A data virtualization server can then transform the data and extend it with more descriptive data from the data warehouse to make it meaningful for the users. This way, all four limitations are overcome elegantly.

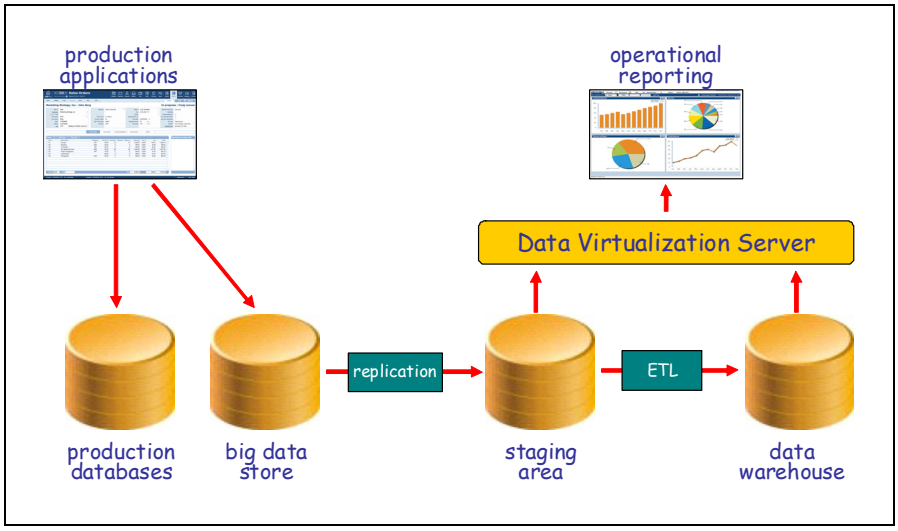

Figure 12 *Replicating coded data from the big data store to the data warehouse environment*

Note: Replicating *from* a big data store is technologically complex and is at the time of writing not yet available in the market. Vendors may provide this solution in the near future.

To summarize, replication can be used to make data from a big data store environment available for operational BI. In addition, data virtualization helps to turn operational data in a form required by the users.

9 Informatica's Data Replication Product

Informatica is well-known for its ETL product called PowerCenter. But Informatica supports the other forms of data integration and data transformation as well: data replication and data virtualization.

In July 2011, Informatica acquired WisdomForce. This vendor had developed a data replication product which is now called *Informatica Data Replication* (IDR). IDR is a highly efficient and scalable data replicator. Besides all the standard features expected from a data replication product, IDR offers the following unique set of features:

- **Heterogeneous deployment:** The product can be used for data replication in an environment where the source and target database servers are different:
    - **Extensive Database Server Support:** IDR supports a wide range of source and target data stores, including all the well-known database servers: IBM DB2, Microsoft SQL Server, MySQL, Oracle, PostgreSQL, and Sybase ASE.
    - **High-End Analytical Platforms:** Specifically for BI systems, as target data stores IDR also supports some of the high-end analytical platforms, including IBM/Netezza, EMC/Greenplum, HP/Vertica, and Teradata. For each platform separately, IDR supports an optimized load form. Because of the large data volumes handled, this is an important feature. Now that more and more organizations are adopting these platforms for developing their data warehouse environment, it's important that data replication tools support these platforms.
    - **Big Data Support:** With respect to *big data* stores, IDR is able to load data into Hadoop big data stores.
    - **Database Agnostic:** Development of the replication specifications is done in a database agnostic way. This makes the whole replication environment database independent. In addition, developers don't have to understand the details of each individual database server, but can focus on the logic of replication.
    - **DDL Replication:** Besides replicating changes to the data, IDR can also replicate changes to database objects, such as tables and columns (DDL changes).

- **High Availability:** To improve availability, IDR can keep on running even if the target system is down. It remembers which data has already been loaded and which data still has to be loaded.

- **Full Range of Configurations:** From a single site, IDR supports replication in any kind of configuration, from simple one-on-one- configurations to complex many-to-many configurations. With respect to many-to-many configurations, data can be replicated from many data sources to many data source. This eases development and maintenance.

- **Console-Based Management:** IDR offers a console-based, developer-friendly, GUI-based interface. The console allows for centrally managing and monitoring the replication processes and is used for entering replication specifications. This interface eliminates the need for developing in low-level scripting languages and thus improves productivity. Scripting is available if a need exists to go beyond the standard functionality, such as adding special-case transformation.

In different ways, their ETL product PowerCenter and IDR are integrated. For example, for certain source systems IDR uses the *change data capture* features of PowerCenter. The expectation is that in the future the products will integrate more.

For completeness sake, Informatica supports *Informatica Data Services* for data virtualization. More on this product in upcoming whitepapers.

10 Three Short Case Studies

This section contains three short case studies of organizations using and benefitting from data replication. All three use Informatica Data Replication (IDR) to support operational BI.

Westlake Financial Services – Westlake Financial Services specializes in the acquisition and servicing of near-prime to subprime automotive retail installment contracts.

Westlake uses IDR to replicate data from their production systems using the Oracle database server to their MySQL-based reporting environment. Replication is done in real time, allowing reports to access operational data.

By using data replication the data reporting latency is reduced from 1 day to 10 minutes. This latency reduction was necessary, because with the old solution business decisions were made based on stale data (one day old). The effect was that in some situations customers were questioned about payments they had already made.

The benefits for Westlake are effectiveness of their payments collection system and increased profitability. In addition, their reporting system became more agile, more focused on responsive and rewarding customer care.

Pharmacy Health Care Provider – The key problem of this large U.S. pharmacy health care provider was that operational data required for business performance decisions, patient claims, and pharmacy management was delayed by as much as 24 hours, even though the service-level agreement often called for data latency to be less than 4 hours.

Every day they capture data relating to more than five millions claims. Their production and data warehouse environment were developed with the Oracle database server. A decision was made to replace the Oracle data warehouse by a Teradata solution, and to replicate data from the Oracle production environment to their new Teradata data warehouse using IDR.

Currently, the solution is transporting approximately 30 tables using IDR. The number of columns per table varies between 30 and 350, with many table volumes exceeding 100 million rows per table. The organization reported that the extraction time from the Oracle database is up to 10 times faster than before.

Their business advantages of using data replication are improved speed of reporting, accelerated decision making, and fast return of data. It also enabled them to respond more rapidly to changing requirements of the health care market, the end-user pharmacy customers, and to the needs of its network of pharmacy locations. Support for payers and patients also improved.

Telecommunications Company – A major Asia-Pacific telco replicates their call detail records from their multi-terabyte Oracle-based operational data store to a data warehouse developed with HP Vertica Analytics Platform in 20-minute billing cycles. The system can now alert customers when they are about to exceed their cell phone minutes in their plan before they incur the additional expense, thereby avoiding bill shock. Customers can now access their records via a portal tied to the Vertica data warehouse.

This solution provides two benefits. First, customer satisfaction has improved increasing retention, and second, revenues have increased as customers realize the advantage of increasing their cell data plans based on their usage patterns.

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.

The last years he has focused on applying data virtualization in business intelligence system resulting in his upcoming book entitled *Data Virtualization for Business Intelligence Systems* which will be released in the summer of 2012.

Rick is chairman of the annual European Data Warehouse and Business Intelligence Conference (organized in London) and chairman of the annual BI event[3] in The Netherlands. He writes for BeyeNetwork.com[4]. He introduced the business intelligence architecture called the *Data Delivery Platform* in 2009 in a number of articles[5] all published at BeyeNetwork.com.

He has written several books on SQL. His popular *Introduction to SQL*[6], published in 1987, 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.

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) or via Twitter (http://twitter.com/Rick_vanderlans).

About Informatica Corporation

Informatica Corporation is the world's number one independent provider of data integration software. Organizations around the world gain a competitive advantage in today's global information economy with timely, relevant and trustworthy data for their top business imperatives. More than 4,100 enterprises worldwide rely on Informatica to access, integrate, and trust their information assets held in the traditional enterprise, off premise, and in the Cloud. For more information, call +1 650-385-5000 (1-800-653-3871 in the U.S.), or visit www.informatica.com. Connect with Informatica at http://www.facebook.com/InformaticaCorporation, http://www.linkedin.com/companies/3858, and http://twitter.com/InformaticaCorp.

[3] See http://www.bi-event.nl/59857
[4] See http://www.beyenetwork.com/channels/5087/articles/
[5] See http://www.beyenetwork.com/channels/5087/view/12495
[6] 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