



CONSULTANCY

Combining Data Streaming with Data Virtualization for Real-Time Analytics

When Every Millisecond Counts

A Technical Whitepaper

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

October 2015

Sponsored by



Copyright © 2015 R20/Consultancy. All rights reserved. Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. or there countries. To view a list of Cisco trademarks, go to this URL: www.cisco.com/go/trademarks. Trademarks of companies referenced in this document are the sole property of their respective owners.

Table of Contents

1	Introduction	1
2	Data Streaming in a Nutshell	1
3	Streaming Data and Business Intelligence	3
4	The Internet of Things is all About Streaming Data	5
5	Characteristics of Streaming Data	5
6	Architectural Aspects of Streaming Data	7
7	The BI Opportunity: Real-Time Analytics of Streaming Data	9
8	Overview of Cisco Connected Streaming Analytics	10
9	Combining Real-Time Analytics with Data In Rest	13
	Simplifying Near Real-Time Analytics	13
	Enriching Classic Reporting And Analytics With Streaming Data	14
	Enriching Real-Time Analytics With Data In Rest	15
	Offloading Streaming Data	16
	About the Author Rick F. van der Lans	18
	About Cisco Systems, Inc.	18

1 Introduction

Introduction – “When every millisecond counts” is the subtitle of this whitepaper. It refers to those BI environments in which, in the blink of an eye, decisions are made based on data generated by events. Such environments for *real-time analytics* analyze the data produced by events instantaneously, they make (automated) decisions, and they initiate an immediate reaction. Generally, the time between the event that produces the data and the reaction is merely a few milliseconds.

In a traditional BI environment it can easily take minutes, hours, and even days, before the data representing a specific event is analyzed, a decision is made, and some action is taken. Therefore, integrating real-time analytics with a classical BI environment is not straightforward.

Data Streaming – Real-time analytics requires *data streaming technology*. This technology is capable of listening to massive amounts of data messages coming from all kinds of events and from all kinds of devices, such as sensors, phones, smart meters, and applications. Data streaming technology is designed and optimized specifically to analyze data instantaneously, make decisions fast, and take actions instantly. Power plants, telecommunication companies, websites, hospitals, credit card companies, and many more organizations deploy this type of technology. Data is generated by some device, machine, or application—the *producers*—and transmitted (streamed) over a bus or network. Other devices, machines, or applications listen to these messages and are instructed to react as quickly as possible based on the data content. The latter ones, the *consumers*, analyze the streaming data real-time.

The Whitepaper – This whitepaper describes data streaming, the relationship with business intelligence, and lists the characteristics of Cisco’s product for data streaming called *Connected Streaming Analytics* (CSA). It also explains how the combination of CSA and Cisco’s data virtualization called *Cisco Information Server* (CIS) can be used to develop BI systems that merge the traditional forms of analytics and reporting with real-time analytics. In other words, combining CIS with CSA allows reporting and analytics on *data in rest* and on *data in motion*.

By deploying CSA and CIS, various approaches exist to combine real-time analytics with classic forms of BI with the intention to simplify, enhance, enrich, or extend the analytical reporting capabilities of one or both. The following four approaches are described:

- Simplifying near real-time analytics
- Enriching classic reporting and analytics with streaming data
- Enriching real-time analytics with data in rest
- Offloading streaming data

2 Data Streaming in a Nutshell

What is Data Streaming? – The term *data streaming* commonly identifies an IT system in which data is continuously transmitted from one system (the *producer*) to another (the *consumer*) which reacts instantaneously on the incoming data. The words “continuously” and “instantaneously” are key here. The term data streaming is used, because producers don’t transmit data periodically or occasionally, but nonstop. A continuous stream of data is produced; comparable to a river. Just as important is the

instantaneous reaction by the consumers. When consumers receive data they act upon it with no delay. It's a *real-time* reaction.

With data streaming it's all about *volume* (the amount of streamed data) and *speed* (the instant response by consumers). In some data streaming systems millions of messages are generated per second and the required reaction is measured in milliseconds. This makes data streaming a very specific type of IT system. Most IT systems store data before it's processed and before decisions are made based on the data. With data streaming, decisions are made before the data is even stored. In fact, in most data streaming systems no time exists to store the data. It would cause too much delay. Figure 1 contains a simple comparison of a data streaming system and a more classic IT system; the difference is storage.

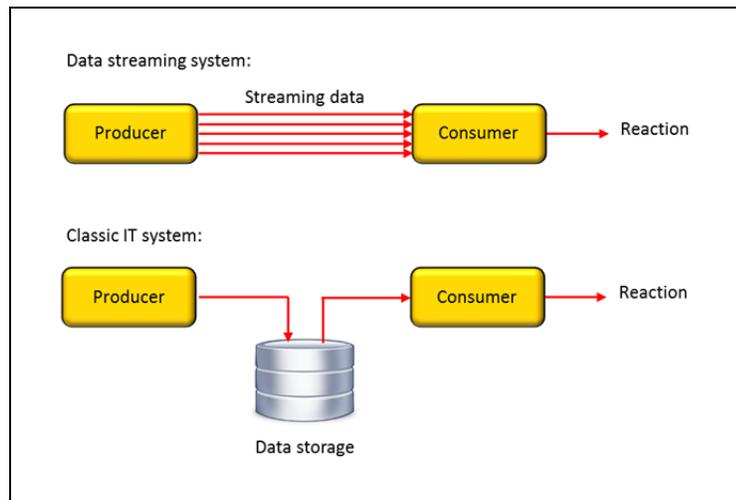


Figure 1 *The top diagram represents a data streaming system in which data is not stored before it's processed by the consumer. The bottom diagram represents a more classic IT system in which data is stored before it's consumed.*

Examples of Data Streaming – Data streaming is almost as old as the IT industry itself. What has changed over time is the amount of streaming data and the speed with which a reaction is required. Over forty years ago the world of *process computing* was already deploying data streaming. For example, factories used simplistic sensor technology that generated data and transmitted (streamed) that data to other devices, such as gauges, to display the state of an engine, a component, or the production of a machine. It allowed human operators to see what was happening live.

Today, due to the advances of hardware and software technology, many more data streaming systems exist, and the volume and speed of streams have increased drastically. The consequence is that currently data streaming is everywhere. Not only large factories stream data, but all kinds of companies. For example, telecommunication companies monitor outages to detect drops in the service level; internet companies monitor every form of usage by website visitors to influence which ads to present or which products to recommend; power plants monitor overheating of components; banks use it with their stock tickers and for real-time fraud detection; distribution centers monitor incoming and outgoing products with their RFID readers; on-line gaming companies deploy it to detect malicious behavior and monitor quality of service; hospitals; offices; and the list goes on.

Data Streaming in Our Daily Lives – People may not be aware of it, but in their daily lives they are responsible for generating numerous, massive data streams. For example, by driving their cars they generate several data streams, such as the speedometer that consumes the data stream coming from the speed sensor and the temperature gauge that presents data coming from the sensor that measures the outside temperature. When the outside temperature is below zero, the latter may even give a signal to

warn the driver that the ground may be frozen and slippery. Both are examples of simple systems in which streaming data is primarily displayed and no actions are taken. That's different with the sensor on the windscreen. If it measures too much rain on the windscreen, it activates the wipers.

But the car is not the only device people use to generate data streams. Smart phones, tablets, and smart watches are continuously producing massive amounts of data that are transmitted to providers. The apps on our phones also stream data continuously. But it doesn't stop with the devices we carry around. When we watch TV, the provider monitors which programs we watch which influences the ads that are shown. Smart energy meters send data to utility companies.

New devices have become available that use streaming data. For example, the company called FitBark (www.fitbark.com/) sells little devices that can be attached to dogs to measure their well-being. This data is sent across the Internet to the data center of FitBark where the data is analyzed and transmitted in a readable form to the dog owner's smart phones so that they can check if the dog is doing well.

Nowadays, almost everything we do is being measured and monitored and many of these streams of data are processed instantaneously.

The Event – Streaming data is always triggered by some *event*. When the event takes place, data is produced and transmitted. Many different types of events exist. For example, an event can be that a specific date and/or time has been reached, a user touches a button on the website, a microsecond has passed and a new measurement has to be taken, a payment is made with a credit card, a customer enters a store, or a car passes a toll booth that automatically takes a photograph of the license plate.

Note: There is always an event that's responsible for generating the data. Don't compare events with business processes. The latter is a much "larger" concept. A business process may comprehend many events.

3 Streaming Data and Business Intelligence

Streaming Data Involves Decision Making – In most data streaming systems the reaction by the consumer application is driven by some *decision process*. For example, telecommunication companies monitor call drops of mobile phones. By analyzing the data stream a decision can be made to offer customers a gift coupon when they experience too many call drops in a particular period of time. Another example relates to factories; when the data stream emitted by a temperature sensor shows overheating of a machine, a consumer application can decide to switch off the entire machine. A well-known example is the weblog data stream indicating the webpages that visitors look at; analysis of this data stream can influence which ads to display.

Conclusion, the part of a data streaming system where results are consumed makes decisions. Some are simple ones, such as, if the temperature measured by the sensor (the producer) is higher than 70F, then send an alarm email to an operator. This decision is based on a simple if-then-else rule that can be evaluated purely on the incoming data. But a decision can also be highly complex. For example, a data stream must first be joined with some other data source and the result is to be evaluated with a statistical model to determine which action to take.

Business Intelligence is About Decision Making – Decision making has always been the domain of business intelligence (BI). This is clear from one of the more accepted definitions of BI by Boris Evelson of Forrester Research¹: “Business intelligence is a set of methodologies, processes, architectures, and technologies that transform raw data into meaningful and useful information used to enable more effective strategic, tactical, and operational insights and decision-making.” In short, BI is about supporting and improving decision making within organizations.

Although the consuming aspect of data streaming contains a decision making process, it has never had a strong tie with business intelligence. Most current data streaming systems have no or a minimal relationship with business intelligence systems. The streaming data is usually not copied to a data warehouse and made available for other forms of reporting and analytics. In fact, process computing and BI have always been very separate worlds. But this is changing. Organizations have discovered that combining streaming data with more classic enterprise data has a *synergetic effect*. One of the first successful application areas of combining the two is the Internet.

What’s Different About Streaming Data for BI? – But why have streaming data and BI always been separate worlds? Let’s look at this from the BI perspective. The BI industry has always felt comfortable with analyzing *stored* data alias *data in rest*, and not so much with streaming data alias *data in motion*. Before data from production systems or websites is analyzed, it’s first stored in a data warehouse or data mart. In fact, almost all the tools available for reporting and analytics can only function when data is stored in some database or file system. The data must first be brought to *rest*.

In the BI industry three assumptions have always been made (almost subconsciously). First, all the incoming data to be analyzed can be stored; second, enough time exists to store the data; and third, if data is stored it can be used instantly and later on. These assumptions don’t always apply to data streaming systems.

First, the amount of data can be just too much or too expensive to store. Imagine that every dog owner in the US buys a FitBark device (see Section 2) and every 10 seconds all these devices transmit one measurement. There are approximately 80 million dogs² in the USA. This produces 8,640 messages of data per dog per day and for all the dogs together the stream of data is 691,200,000,000 messages per day. Even when these messages are short, say 40 bytes long, the new amount of data to be stored per day is still an impressive 27 terabytes. Storing all this data would be a technological challenge and very expensive.

The second assumption relates to the time it takes to store data. Storing one new record with data may only take a fraction of a millisecond, but for some applications this may be too slow. There are critical data streaming systems in which every second counts. For example, in the trading world, reacting to an event on the stock market a few milliseconds too late can mean the difference between losing or winning millions of dollars.

The third assumption relates to storing data for a long period of time. In many organizations, once data is stored in a data warehouse, it is never thrown away. It’s a once-in-never-out system. No mechanisms are in place to temporarily delete old and obsolete data. One of the reasons is that organizations don’t know

¹ B. Evelson, *Topic Overview: Business Intelligence*, November 21, 2008.

² Statista, *Number of Dogs in the United States from 2000 to 2015*, August 2015; see <http://www.statista.com/statistics/198100/dogs-in-the-united-states-since-2000/>

whether this old data may become useful for analysis later on. Streaming data, on the other hand, is commonly not stored for a long period of time, because some of it has no value after it has been analyzed. Storing a highly aggregated form of the data may be sufficient. Take a sensor that measures the temperature of a machine component every 10 seconds. Is it really useful to store all these individual measurements? It probably isn't useful when for 24 hours straight the temperature remains unchanged. Storing streaming data for long periods of time is not always the right option.

Conclusion, data streaming systems are different from the traditional BI systems. Not all the streaming data is stored, there is not always enough time to store the data, and data is not always used later on.

4 The Internet of Things is all About Streaming Data

Big data is immensely popular in the IT sector, and rightfully so. With respect to popularity, the *Internet of Things* (IoT) (or the *Internet of Everything*) is closely following in its footsteps. In the classic Internet most of the communication takes place between humans and machines; people click on webpages which results in messages that are transmitted to applications and machines. With the IoT, devices, such as sensors, network interfaces, phones, computers, and so on, communicate amongst themselves. They generate data that is transmitted over the Internet and is received and processed by other devices and applications somewhere else on the Internet. For example, smart thermostats installed in homes periodically emit energy consumption data that is transmitted across the Internet to an application of a utility company which uses it to automatically distribute energy across the network based on current consumption. The FitBark example is also a typical IoT system: the devices attached to the dogs communicate across the Internet with mobile phones. The “things” are turning into “smart things.”

The IoT is not something of the future, it's already there. Gartner³ forecasts that 4.9 billion connected things will be in use in 2015, and this number will reach 25 billion by 2020. Especially the manufacturing, utilities, and transportation industries will be the top three verticals deploying the IoT.

Most IoT systems are data streaming systems. The general expectation is that due to the increasing popularity of IoT, the number of data streaming systems will explode. So, maybe today streaming data may be somewhat special, but it won't be any more within a few years. The IoT will become a key initiator for organizations to develop data streaming systems.

5 Characteristics of Streaming Data

The data processed in streaming environments is generated by devices, machines, and sensors and has very specific characteristics that can be a little different from those of more classic data stores. This section describes some of these differing characteristics.

Short Record Length – Typically, the records of machine-generated data streams are short. They are kept short to speed up transmission times and, if relevant, to speed up storage. Generally, these short records contain a timestamp indicating when an event took place; some identification of the machine, sensor, or

³ Gartner, *Gartner Says 4.9 Billion Connected “Things” Will Be In Use In 2015*, November 11, 2014; see <http://www.gartner.com/newsroom/id/2905717>

phone that produced the data; an indication of the location (of the machine, sensor, or device); and one or more measurements.

Note that not all the streaming data is short. For example, security video cameras send out continuous streams of video and photographs taken continuously from people entering an airport (to be used by face-recognition algorithms). These are massive data streams. Still, most data streams contain short records.

Simple Structure – The structure of the records in a data stream is simple. To illustrate this, here are two records from a stream produced by a sensor:

welluid	qtime	xmlmints	xmlmaxts	length long fiber	temp before	temp after	temp delta	temp min	temp max	temp range
wellC	2015-05-14 12:03:59.999999	2015-05-14 09:41:49	2015-05-14 12:04:08.205	499.633	16.1331	20.9771	4.844	16.1331	20.9771	4.844
wellC	2015-05-14 12:04:09.999999	2015-05-14 09:41:49	2015-05-14 12:04:08.205	499.633	16.1331	20.9771	4.844	16.1331	20.9771	4.844

To minimize network traffic, in some systems multiple measurements are merged into one record. For example, one record may hold ten consecutive measurements. This is sometimes referred to as a *repeating group*.

The structure of streaming data is commonly highly stable. Not every record differs in structure nor does the structure change frequently.

Cryptic Data – Streaming data is usually cryptic. For example, it doesn't contain the names of the machines or sensors generating the data, but codes. This makes analyzing the data somewhat complex. First, it must be translated to something that can be understood by humans. Even the measurements may have to be processed before they can be interpreted by human analysts. One of the reasons to use cryptic codes is to keep the records short.

Classic production systems may contain cryptic codes as well, but in most situations the explanations of these cryptic codes (also known as reference data) are available in the same system. Normally, the data becomes understandable by simply joining the table containing the cryptic codes with the one containing the explanations. A data stream doesn't contain the explanation. A join is required with data from another system that contains the explanations.

Streaming Data is Clean – In classic BI systems much effort is dedicated to cleansing the data stored in production systems. One of the reasons that cleansing is required is that most of that data is entered manually and may contain typos and other types of errors. This does not apply to machine-generated data. In a way, that data is always correct because it's directly generated, not entered manually. Although, there can be some problems with its "correctness". For example, a sensor may be defect and may emit skewed measurements. It may also be wrongly installed. For example, attaching a living room thermostat to a radiator returns temperature measurements that are too high. These measurements are not representative for the average temperature in the living room. Some devices may also occasionally produce values that are unrealistic. Such *outlier values* can be removed by the listeners of the data streaming technology.

Missing Data – When the frequency of incoming data is high, records may get lost due to the order in which they arrive. Due to technical reasons records flowing from a producer to a consumer may arrive in the wrong order. Consumers can decide to discard all the ones that do not arrive in the right order, but that leads to *data loss*. In this case, records must be buffered and reordered to increase the systems’ tolerance to the order in which data arrives. In other words, these mechanisms tolerate a specified amount of “slack”, which is basically a bound on the degree to which input can be unordered.

6 Architectural Aspects of Streaming Data

The Three Steps of Streaming Data – On a high architectural level, data streaming systems logically consist of three steps; see Figure 2. There is always some *listener* that inspects messages transmitted over an enterprise service bus, a message queue, or some other data transportation mechanism. In most cases, the data is transmitted in the form of messages. The listeners select the relevant messages that adhere to specific conditions. So, listeners may slim down the stream of data to be processed by the preceding steps. Depending on the situation and the technology in use, there may be several listener processes at work to deal with the massive streams of incoming data.

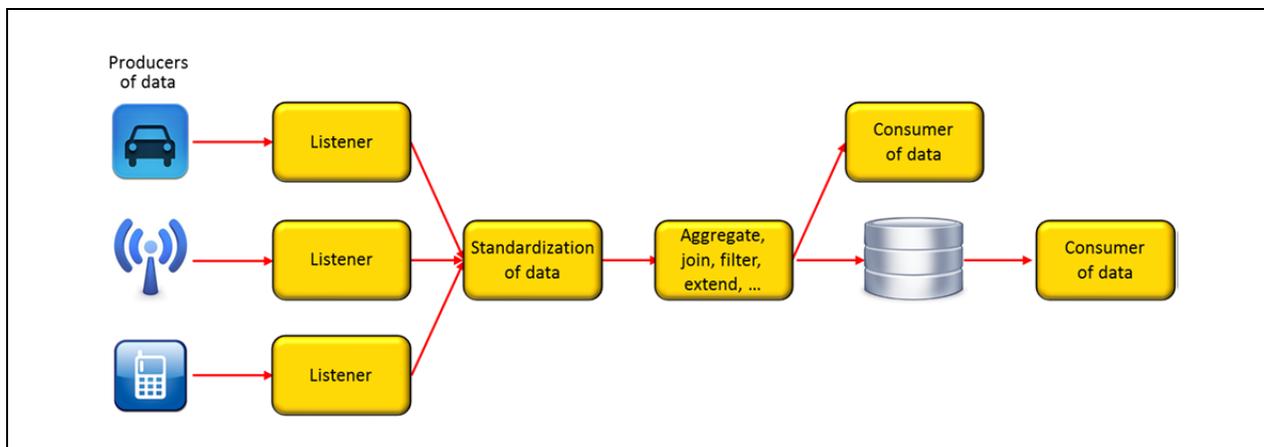


Figure 2 A data streaming system logically consists of three steps.

Messages from different producers can have different formats. For example, one may use a comma-delimited structure and another XML or JSON. Therefore, in the second step, the selected messages are transformed to a standard format. This step doesn’t change the data, but only the format. This simplifies the work to be done in the next step.

In the third step, the messages are processed in a more advanced way. Data is filtered, aggregated, split, extended, and so on. Also, data from multiple streams may have to be merged. In addition, data in motion may have to be joined with data in rest that is stored in some data source.

When the third step finishes processing the data, the data is placed on some transportation mechanism for consumption by another application or is stored in a database for future use. An example of the former is when data coming from a sensor installed in some high-tech machine has to be aggregated over time in step 3. Based on these aggregations this step determines whether a machine component is getting overheated, and if so, an email must be transmitted to a controller in the factory. So, no data is stored. An

example of the latter is a system in which every minute the measured temperature of a machine component is stored in a database. By storing the streaming data the temperature's history over time can be studied and patterns can be identified. Is the component slowly heating up or are there occasional temperature peaks? Next, an analysis is needed to study the increasing warmth over a one-year time frame. In other words, how did the temperature increase one year ago, and how does that compare to this year? Is there a difference? So, data may be used right away by some other system or is stored in a database.

Time is of the Essence – In traditional BI environments the time between events producing data and the time it's consumed and analyzed, can range from minutes to days. This is not the case with streaming data. Very often the data must be analyzed within microseconds and sometimes even milliseconds, or else the reaction is too late. Take the stock trading world, when a response is not instantaneous, it's worthless.

Automated Decisions – In most BI systems the human business user analyzes the data and makes decisions. In many data streaming systems, decisions are made automatically by some hardware or software component. Take, for example, a sensor that measures continuously the size of pressed sheet-metal. If, within a particular time period, the thickness of the sheet-metal falls outside acceptable boundaries, the machine must be stopped. This process embeds real-time automatic decisions.

Eventually, human users are involved in this process, but not real-time. The rule to decide what's too thick or too thin has to be made in advance. That's human work, but the real-time decisions are made automatically. This is the same with face recognition software that monitors airport passengers. Decisions are made automatically by the software when it thinks it recognizes some passenger, while the algorithm that decides whether a face is identical to one in a predefined list is developed in advance.

Storing Aggregated Data – When streaming data must be stored, it can be sufficient to store an aggregated form of the data. Imagine that in a large office building the sun blinds outside are automatically closed based on the amount of sunshine outside and the temperature in the rooms. For this, every 5 seconds the intensity of the sunshine and the indoor temperature are measured. This data can be used to control the sun blinds, but does all this data has to be stored as well? Storing 24 hours x 60 minutes x 12 measurements x 2 sensors leads to 34,650 records of data per day. Storing an aggregated form of the data, for example average values per 10 minutes, is probably sufficient for future use of the data. Storing brings it back to 288 records.

In many data streaming environments the decision is made to store aggregated data. But there are examples where this is not an option. For example, in a medical environment in which a sensor measures possible peaks in patients' heart beat rhythms, storing aggregated data is not recommended. The aggregated data may hide some of the heart beat peaks.

Short Life Span of Data – Most of the standard enterprise data, such as financial and medical data, is stored for years. Most of the streaming data doesn't have to be stored for a long time. Keeping it for a few weeks or months is often long enough. So, in general the life span of streaming data is relatively short. In these systems it's therefore important that mechanisms are in place that delete obsolete data.

7 The BI Opportunity: Real-Time Analytics of Streaming Data

The Synergetic Effect – A lot of expertise is available in the BI industry to analyze *data in rest*. Many know how to, for example, design data warehouses and datamarts, how to tune ETL processes, how to process dirty data, and how to design insightful reports. However, analyzing streaming data is for many unknown and there is a little or no expertise in this field. This is a missed opportunity, because organizations that have discovered how to bring these two types of data together show that it can have a *synergetic effect*. By enriching streaming data with enterprise data, and vice versa, the analytical capabilities of an organization are extended.

But what exactly is this synergetic effect? Imagine a manager responsible for the day-to-day transport operations within a large retail company. The classic BI system gives insight into past transport accomplishments, such as the percentage of trucks that have delivered products on time and the average load per truck. It may even give information about the available transport capacity versus the required capacity for the coming quarter. Still, for the manager it would be beneficial if he also gets real-time data on the locations of all the trucks for route optimization, whether trucks have engine problems to proactively reset delivery times, and so on. This real-time data can come from a data streaming system. Combining the more historical data from the BI system with the streaming data creates a more complete picture of what has happened, is happening, and may happen. That's the synergetic effect.

In the financial world, combining data in rest with data in motion has been daily practice for a long time now. We have all seen these pictures and films in which stock traders watch multiple screens attentively on which real-time data is presented together with data in rest. One screen may be showing real stock ticker data (data in motion), another may contain a full day overview of how stock indices as NASDAQ, Dow Jones, and the S&P 500 are doing, and the third may contain a long term indication of the value of several investment portfolios. Combining the most current situation with the short and long term trends is indispensable for these traders.

A Classic BI System: Store, Store, and Store – The first solution that comes to mind to combine both types of data is to regard each data streaming system as a data source in a classic BI system. A classic BI system consists of a chain of databases and transformation processes; see Figure 3. On the left are all the production databases that contain source data and on the right are all the systems for reporting and analytics. With several transformation steps (ETL) the structure and content of the data is transformed gradually to make it suitable for analysis and reporting. In this system, data is stored multiple times.

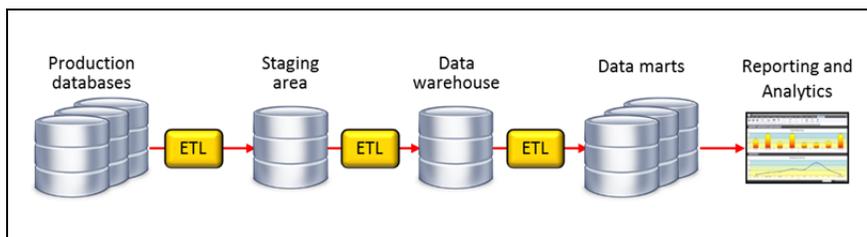


Figure 3 *Classic BI systems store data multiple times before it's used for reporting and analytics.*

To bring the two forms of data together can be done by considering the data stream as some source system; see Figure 4. The streaming data is first loaded in the staging area, then copied to the data warehouse and onwards to the datamarts. Subsequently, the data in motion is available in the same way

as the data in rest. Bringing the two data forms together is purely a matter of joining a couple of tables from the data mart together.

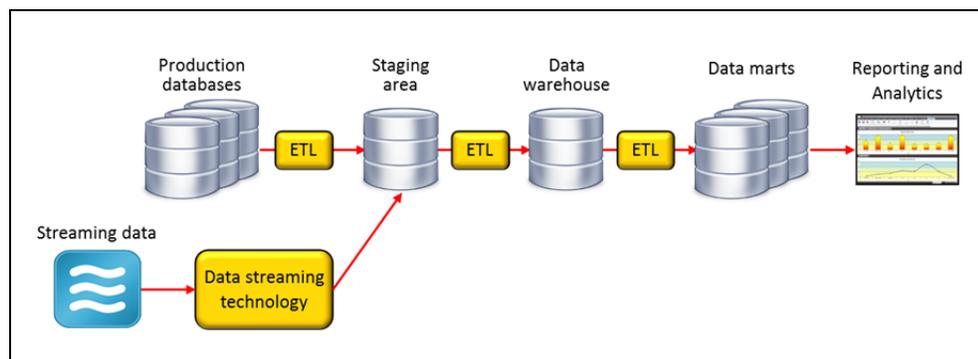


Figure 4 *Classic BI systems in which a data streaming system is regarded as one of the data sources.*

This looks like a good solution, but it's not the right architecture for analyzing streaming data real-time. In this architecture it takes quite some time before the generated data is available for analysis. Now, it doesn't have to take minutes to push data through this chain of databases, but it won't be done in microseconds either. Too much storing is going on before the data can be analyzed. When the data in motion has been copied several times, it's not real-time anymore. This delay can be disastrous for some business users. Another architecture is required.

Note: It doesn't mean that this classic architecture cannot play any part when analyzing streaming data. It can and it must, because it contains most of the data in rest. An architecture is required that brings the two forms of data together in such a way that it can show real-time data in combination with the data in rest. Such an architecture can be developed by deploying a product for streaming data and one for *data virtualization*, and those two must be able to work together. This architecture is described in Section 9. But first, the next section describes Cisco's product for data streaming.

8 Overview of Cisco Connected Streaming Analytics

This section presents a high-level overview of Cisco's product for streaming data, called *Cisco Connected Streaming Analytics* (CSA).

Introduction to CSA – CSA is a SQL-based engine for processing and analyzing streaming data. The product was initially developed by a company called *Truviso*. Truviso was founded in 2006 by Professor Michael J. Franklin and Sailesh Krishnamurthy both working for the University of UC Berkeley. Michael Franklin is involved in a number of projects that have attracted worldwide attention: Spark, Shark, GraphX, and BDAS. To quote Wikipedia, Truviso “enables analysis of streaming data, including queries that combine those streams with other streaming data or with historical/staged data.”⁴ It's not strange that Truviso was “born” in Berkeley. This city has always been a hotbed for innovative data storage and processing technologies, such as Ingres and PostgreSQL in the 1970s and 1980s, and currently Hadoop and Spark. Truviso was acquired by Cisco on May 4, 2014. Right after the acquisition, the product was initially called *Prime Analytics*, but the current name is CSA. CSA has been designed to be scalable. For example, one customer uses CSA to process over 250,000 messages per second.

⁴ Wikipedia, *Truviso*, September 2015; see <https://en.wikipedia.org/wiki/Truviso>

The Architecture of CSA – The architecture of CSA is presented in Figure 5. The two key modules making up CSA are *TruLink* and *TruCQ*. *TruLink* is responsible for listening to all types of messages and for transforming all these messages to a standard format. Compared to Figure 2 in Section 6, *TruLink* is responsible for the first two steps: listening to incoming messages and standardization of data. *TruLink* is a dedicated program running on a Java application server.

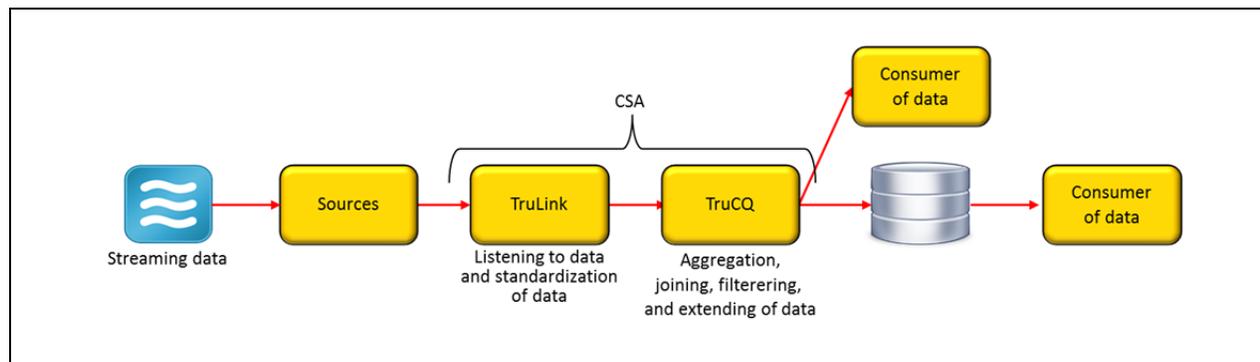


Figure 5 *The two modules of CSA: TruLink and TruCQ.*

*TruLink*⁵ supports a wide range of formats and protocols that it can intercept:

- A variety of network types, including IP next-generation networks (NGNs), mobile, video.
- A variety of network devices/elements: routers, switches, data center servers, and more.
- A variety of data types available from the network, including user location, user device IP address, authentication, authorization, and accounting (AAA) information, and more.
- NetFlow, syslog, NBAR2, EDR, and Extensible Messaging and Presence Protocol (XMPP) data to bring multivendor data into the system.

TruCQ is the brains of the product. Here all the incoming data is processed, analyzed, and stored in a database or send onwards to another system. *TruCQ*, is developed by taking the source code of the popular open source database server *PostgreSQL* and by modifying the internal source code to support streaming data as a first class citizen. It's currently based on Version 9.4 of *PostgreSQL*. The letters CQ stand for *Continuous Query*. In *TruCQ* all the aggregations, joins, filters, and so on, are specified. Because *TruCQ* is developed with *PostgreSQL*, it uses SQL as language to specify filters and aggregations.

In general, there are two use cases for CSA. Data can be readied for *immediate consumption* or for *delayed consumption*. Immediate consumption means that incoming data is processed and immediately placed on some message bus to be used by another system. The processing can involve adding extra derived values, removing irrelevant values, joining streams together, adding extra date and timestamps, replacing codes by meaningful names, and so on. Delayed consumption means that the consumer wants to see the data in some aggregated form. This can be an aggregation over time, for example, the total number of credit card payments per minute or an aggregation over some attribute. SQL is perfect for specifying all these operations.

⁵ Cisco; *Cisco Connected Streaming Analytics Datasheet*, September 2015; see <http://hcsdemo.com/c/en/us/products/collateral/cloud-systems-management/prime-analytics/datasheet-c78-732218.html>

CSA allows execution of all the standard PostgreSQL functions on streaming data. Also, in TruCQ the user-defined function capability can be used. Organizations can develop their own advanced algorithms as user-defined functions that they can apply on streaming data.

Streams and Continuous Queries – What’s special about CSA is the *stream* concept. Streams are very much like tables, they also have columns, the columns have data types, and they contain a set of records with similar data structures. However, an important difference is that the content of a stream does not contain a *finite* set of records. The reason is that a stream is connected to a source system that keeps transmitting data. The content of a stream is defined by specifying a query. This query defines how the incoming data from the source is processed. The full SQL language can be used to filter data, aggregate data, join streams and tables, combine columns, add derived data, and so on. When data is received from the source it is processed based on the query defined within the stream. In other words, the stream is constantly emitting data according to the query specified. Because of this processing style, the query making up the definition of a stream is called an *always-on query* or a *continuous query*. In a way, the query of a stream operates like a fishnet that has been strapped across a river, the water flows through the net, small fishes may even swim through, but large fishes get caught.

To receive that emitted data, applications have to open a *cursor* on the stream. A cursor on a stream works the same way as a cursor on a table. However, since the result set of a stream is infinite, applications need to keep checking if new results are available. Even if an application never checks the cursor again, the engine still generates results. When applications are slower in asking for new records than that records end up being processed by the stream, records are temporarily buffered. CSA keeps returning records to the applications endlessly until the query is stopped.

Applications and tools that need to process CSA streams must support cursors to be able grab the streaming data. Although most tools for reporting and analytics can access most SQL database servers including PostgreSQL, they don’t support cursors. This means that developers have to work with languages such as Java and C to retrieve the streaming data.

Splitting, Aggregating, and Joining Streams – In CSA, multiple queries by different applications can be invoked on the same incoming data stream. The effect is that every record from the incoming stream is processed by all the queries on that stream. In other words, the stream is split into multiple output streams. When splitting a stream, each query can apply different operations on the data: one can aggregate the data, the other can use some filter to look for specific records, and the third can add derived data to the stream.

Multiple incoming data streams can also be joined to form one outgoing data stream. Joining streams together is like joining tables together. In most cases, the streams are joined based on the timestamps of the respective two streams. Because CSA is developed with PostgreSQL, data in rest, such as reference data, stored in the PostgreSQL database can be joined together with the streaming data. For CSA this is merely a join of a table with a stream.

Near Real-Time Analytics – Continuous queries offer real-time analytical capabilities, because the data is analyzed before it’s stored. Many systems don’t need that speed of reaction. Some can live with a near real-time reaction. This can be accomplished in CSA by streaming the incoming data first into a table, and afterwards the data is analyzed with more traditional queries. The speed with which these records can be inserted depends on the hardware configuration, the workload on the machine, and other parameters, but it’s fast, it’s still only milliseconds. Therefore, working with stored data is referred to as *near real-time analytics*.

The Importance of Foreign Tables – CSA is able to write streaming data to the PostgreSQL database, to Hadoop files, and to *foreign tables*. With foreign tables data can be written to tables in other SQL databases. For example, a foreign table can be defined in CSA that points to a table in, for example, Oracle or SQL Server. When a query is executed on the foreign table, the data is retrieved from the other database. The foreign table concept makes it possible that views defined in CIS can be accessed by CSA as well. Such views can then point to other SQL database servers. This foreign table concept enriches the list of data sources that CSA can access, and thus enriches what can be done with streaming data. It also makes it possible to integrate CIS and CSA more closely.

9 Combining Real-Time Analytics With Data In Rest

By deploying CSA and CIS, various approaches exist to combine real-time analytics with classic forms of BI with the intention to simplify, enhance, enrich, or extend the analytical reporting capabilities of one or both. The following four approaches are described:

- Simplifying near real-time analytics
- Enriching classic reporting and analytics with streaming data
- Enriching real-time analytics with data in rest
- Offloading streaming data

Simplifying Near Real-Time Analytics – Data stored in CSA databases can be used for near real-time analytics and reporting. For example, dashboards can be developed that access and analyze data stored in a CSA database. Such dashboards would show near real-time data. Users may not even see the difference between presenting results of real-time analytics (on streaming data) and results of near real-time analytics (on stored data). The difference may be milliseconds and therefore “invisible” to users.

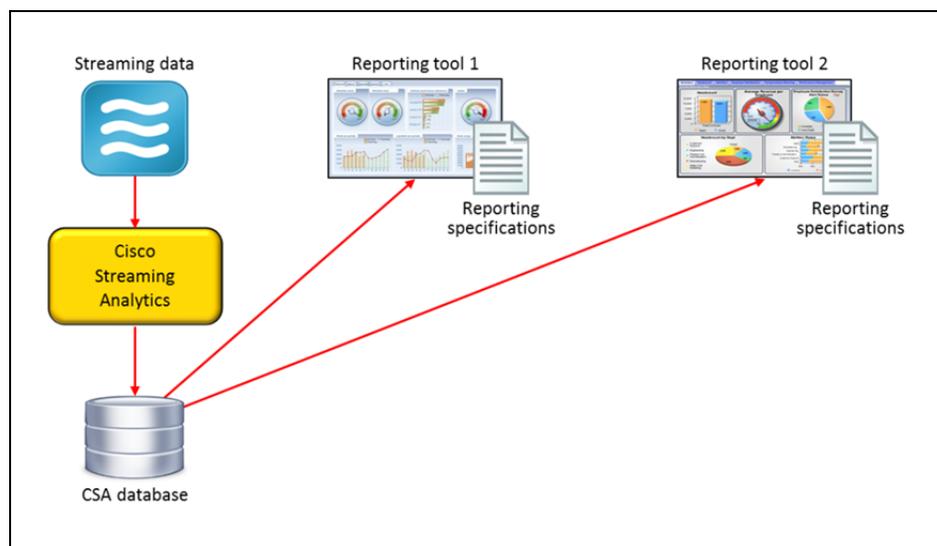


Figure 6 *Allowing reports to access the CSA database directly can be complex and may lead to inconsistent reporting.*

Most existing reporting tools can be used to retrieve data from CSA databases, because most of them allow access to PostgreSQL databases. However, the logic to develop such reports can be complex. One of the reasons is that the data stored by CSA is somewhat cryptic and the data structure is not optimized for reporting but for fast storage; see Section 5. Another disadvantage is that if different reporting tools are

needed to analyze and present data, specifications to access and transform the data for the reports are duplicated across all these tools; see Figure 6. This makes development and maintenance expensive. And when the specifications are not kept in synch, it can lead to inconsistent reporting results. Another potential problem may be that the reporting tools generate so many query requests on the CSA database that this query workload interferes with the storage of streaming data by CSA itself.

By placing CIS in between the database and the reporting tools, views can be defined that present the data in a more practical way; see Figure 7. If multiple tools are used, specifications don't have to be duplicated. In addition, CIS' caching mechanism can be used to minimize access to CSA databases. Note that this cache must be refreshed frequently to present near real-time data. Another advantage relates to the publishing of data. CIS can make the data stored in CSA available via many different APIs and languages.

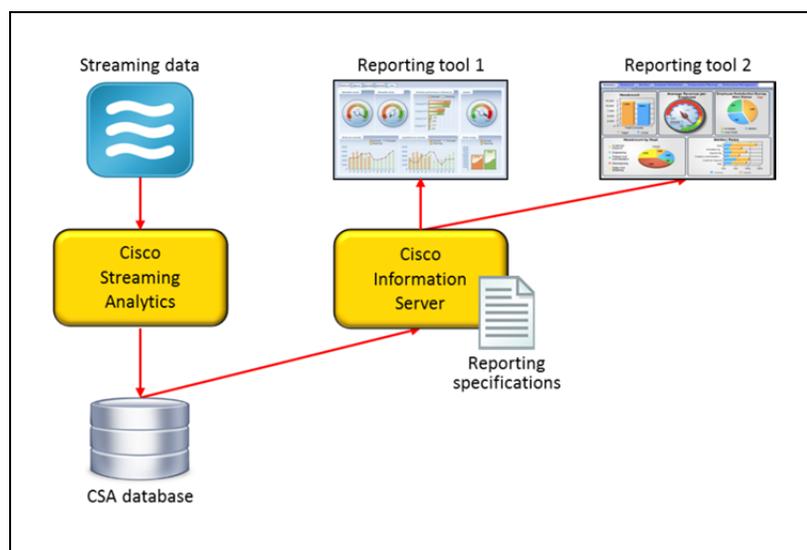


Figure 7 CIS can be used to simplify the development of reports on streaming data. CIS also reduces the need to duplicate reporting specifications.

Data that may be incorrect, such as outliers, may be cleansed in CIS by defining logic in views that removes these outliers or normalizes the values by replacing them by average values. There are several approaches for cleansing the data in view definitions, see the book *Data Virtualization for Business Intelligence Systems*⁶.

Design Guideline: When reports access streaming data through CIS, there are two places where data can be aggregated or processed: in CSA or in CIS. When CSA aggregates the data, storage is minimized, but the reporting capabilities are limited and it requires processing by the CSA engine which may slow down the streaming a little or may lead to the investment in more processing power. If CSA doesn't do any form of processing or aggregation, CIS can do it for the reports. Factors influencing this decision are the speed and volume of the streaming data, the required I/O workload on the CSA database, and the query workload by the reports.

Enriching Classic Reporting and Analytics With Streaming Data – As indicated in Section 7, combining traditional data with streaming data increases an organization's analytical capabilities. For example, sales data of mobile phone subscriptions stored in a data warehouse can be joined with streaming data on the usage of apps. Or, social-demographic data on customers together with that same streaming data allows analysis

⁶ R.F. van der Lans, *Data Virtualization for Business Intelligence Systems*, Morgan Kaufmann Publishers, 2012.

of app-usage based on family size, the average age of children, and the home address (not the location of the phone).

Figure 8 shows how the two tools can work together. Through a JDBC driver, CIS can access all the data stored in the CSA database. Note that the data CIS delivers to the reports in this situation is near real-time, because it's first stored by CSA.

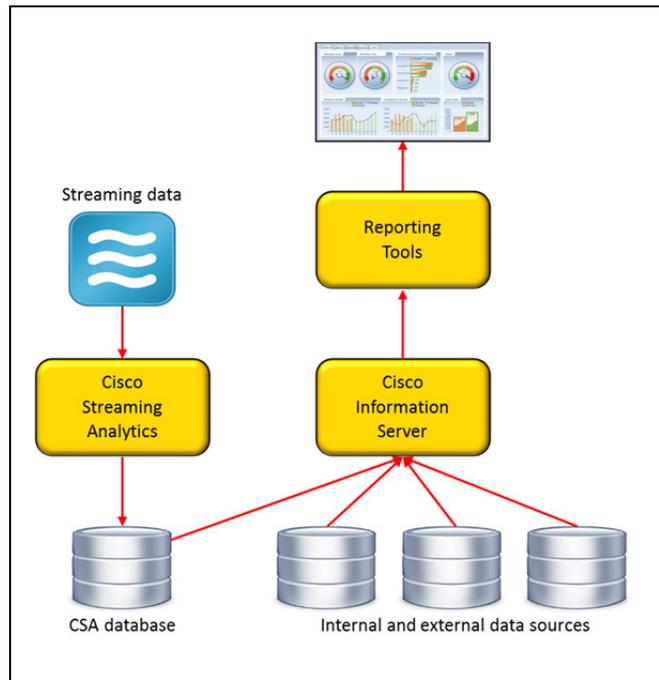


Figure 8 CIS can be used to combine the streaming data coming from CSA with internal and external data sources and thus enriching classic reporting and analytics.

Enriching Real-Time Analytics with Data In Rest – Analyzing streaming data only rarely ever paints a full picture. The key reason is that most data streams don't contain all the required data needed for analytics. More data is needed and most of it has been gathered by organizations in their classic IT systems and data warehouses. For example, a data stream with weblog data may show how customers navigate a website, but it doesn't tell how loyal these customer are. That type of data is probably stored in the CRM system. Or, a data stream with tweeted complaints of particular customers doesn't indicate what the sales figures are for those same customers, nor how many ordered products they returned.

By having CSA access CIS views as foreign tables (see Section 7), the analytical query capabilities of CSA are enriched; see Figure 9. Streaming data can be joined with almost any kind of data source.

All the views defined in CIS can become foreign tables in CSA. Because CIS supports features to transform hierarchical data structures to flat table structures, to transform SQL queries to calls to web services using parameters, it allows CSA to access any kind of data source. CIS supports SQL on a wide range of data sources, including:

- Almost any relational database server, including DB2, Microsoft SQL Server, MySQL, IBM PureData System for Analytics, and Oracle.
- XML documents, MDX databases, flat files, spreadsheets, and other non-relational data stores.
- Data stored in proprietary applications, such as Oracle, Salesforce.com, and SAP. This is supported via the product *Composite Application Data Services*.

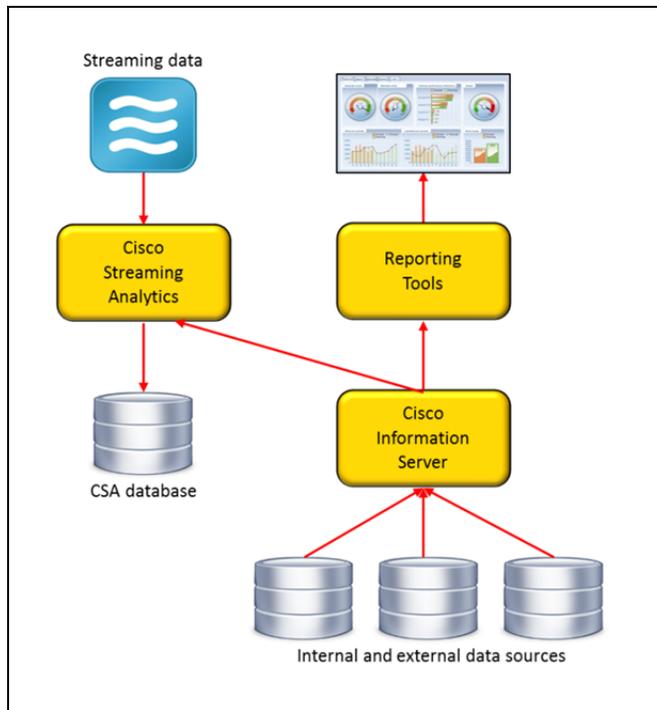


Figure 9 CIS can be used to federate the streaming data with internal and external data sources.

CSA's foreign table technology is not limited to queries. Streaming data can be inserted in data sources accessed through foreign tables. So, streaming data can be inserted into data sources via CIS. This allows that data is inserted straightaway into a high-end database platform without the need to offload the data. This shortens the time that this data becomes available for reporting and analytics.

Offloading Streaming Data – Although CSA offers a wide range of analytical capabilities, it isn't a platform for extensive analytics on the stored data. Its strength is much more in analysis of data in motion, in other words in analysis of data before it's stored. In fact, a heavy analytical workload on the CSA database will undoubtedly interfere with the storage of streaming data. In this case, CIS is the preferred platform for federating data from multiple data sources to supporting analytics.

The PostgreSQL database server is a mature product and can handle large amounts of data. Still, practice has shown that the product works well when the database doesn't exceed approximately 6 terabytes of data. When regulations or specific reporting needs require streaming data to be kept for a long period of time, resulting in a large database, it's recommended to offload data to another database server.

CIS can help to offload the data. Periodically, CIS can invoke an ETL process to copy the data from the CSA database to another database and to remove it from the CSA database after the data has been copied; see Figure 10. CIS can write the offloaded data to many different SQL databases, file systems, Hadoop, and so on. As an example, the whitepaper *Transparently Offloading Data Warehouse Data to Hadoop using Data Virtualization*⁷ describes in detail how CIS can be used to offload data from a SQL database to Hadoop. In the process of data offloading, CIS can extend and enrich data. For example, it can aggregate data, denormalize it, and it can join it with data from other data sources. Note that the offloaded data does not represent the real-time situation anymore.

⁷ R.F. van der Lans, *Transparently Offloading Data Warehouse Data to Hadoop using Data Virtualization*, November 2014; see <http://www.cisco.com/web/services/enterprise-it-services/data-virtualization/documents/hadoop-data-virtualization.pdf>

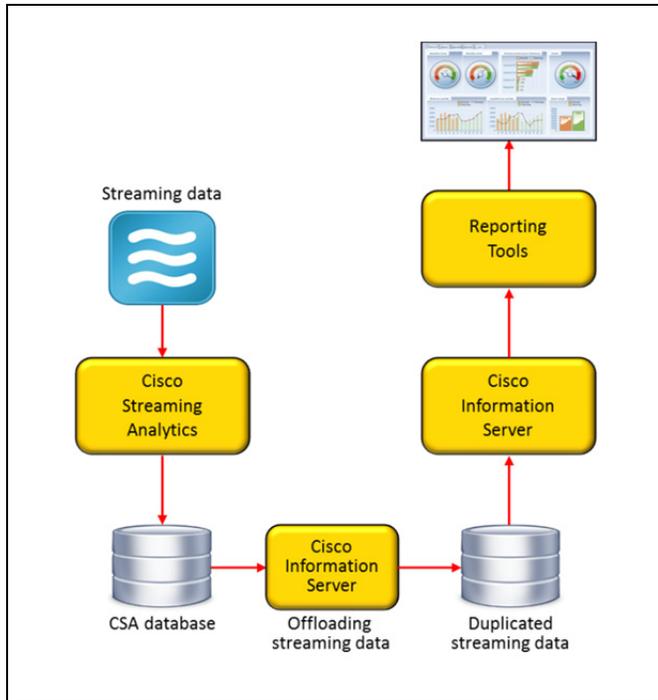


Figure 10 CIS can be used to offload the streaming data to any other data source and make it available for reporting.

Offloading data can also be done with classic ETL products, but the advantage of using CIS is that besides doing the offloading it can make the data easily available for reporting and analytics by defining views on the top of the tables in the database that contains the offloaded streaming data.

As indicated in Section 5, some streaming data may be missing records due to being out of order and the data may be flawed. When it concerns measurement data, stored procedures can be developed in CIS to fill up the missing data by applying interpolation algorithms. These stored procedures can be invoked when the data is being copied.

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, big data, 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 Techtarget.com⁸, B-eye-Network.com⁹ and other websites. He introduced the business intelligence architecture called the *Data Delivery Platform* in 2009 in a number of articles¹⁰ all published at B-eye-Network.com. The Data Delivery Platform is an architecture based on data virtualization.

He has written several books on database technology. Published in 1987, his popular *Introduction to SQL*¹¹ 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 book¹² *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).

About Cisco Systems, Inc.

Cisco is the worldwide leader in helping companies seize the opportunities of tomorrow by delivering the amazing results that come from connecting the previously unconnected. Cisco is empowering countries, cities, industries and businesses around the globe to move faster in order to keep pace with digital transformation and the Internet of Everything.

Cisco Information Server is agile data virtualization software that makes it easy for companies to access business data across the network as if it were in a single place. Cisco Connected Streaming Analytics delivers predictive, actionable insight from high-velocity streams of live data from multiple sources, helping to enable real-time governance and immediate actions.

For more information, please visit

<http://www.cisco.com/c/en/us/products/cloud-systems-management/data-analytics/index.html#~overview>

⁸ See <http://www.techtarget.com/contributor/Rick-Van-Der-Lans>

⁹ See <http://www.b-eye-network.com/channels/5087/articles/>

¹⁰ See <http://www.b-eye-network.com/channels/5087/view/12495>

¹¹ R.F. van der Lans, *Introduction to SQL; Mastering the Relational Database Language*, fourth edition, Addison-Wesley, 2007.

¹² R.F. van der Lans, *Data Virtualization for Business Intelligence Systems*, Morgan Kaufmann Publishers, 2012.