R20
CONSULTANCY

Developing a Data Delivery Platform With Informatica Data Services

A Technical Whitepaper on Next Generation Data Virtualization

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

February 28, 2011

Sponsored by
INFORMATICA®
The Data Integration Company™

Table of Contents

1. Summary … 1
2. Business Intelligence Trends … 2
3. What is the Data Delivery Platform? … 4
4. Advantages of the Data Delivery Platform … 5
5. Data Virtualization, Data Federation, and Data Integration … 8
6. Open versus Closed Federation Servers … 10
7. Federation Servers offer On-Demand Transformation … 11
8. On-Demand Transformation and Cleansing … 14
9. Application Areas of Federation Servers … 15
10. What is Informatica Data Services? … 16
11. Informatica Data Services Under the Hood … 18
12. Defining Virtual Tables … 18
13. Defining Physical Data Objects … 22
14. Defining Data Object Mappings … 24
15. Processing the Mappings by Translating them into SQL … 26
16. Sharing Specifications by Stacking Virtual Tables … 27
17. Integrated and On-Demand Data Profiling … 29
18. Collaborative and On-Demand Data Cleansing … 31
19. Developing Virtual Data Marts … 33
20. Transforming XML Documents and Spreadsheets to Tables … 35
21. Keeping Track of All Relationships … 38
22. Exposing Virtual Tables as Web Services … 38
23. Caching Virtual Tables … 41
24. Optimization Techniques for Accessing Foreign Data … 43
25. Security Features … 47
26. Inserting, Updating, and Deleting Data … 48
27. Technical Advantages of Informatica Data Services … 48
28. Business Advantages of Informatica Data Services … 50
29. Case Studies … 51
About the Author Rick F. van der Lans … 55
About Informatica Corporation … 55

1 Summary

The Data Delivery Platform *(DDP) is a modern architecture for developing business intelligence systems where data consumers, such as reporting and analytical tools, are decoupled from data stores. This whitepaper describes how to develop such a business intelligence (BI) architecture using Informatica's new data integration product* Informatica Data Services*. The concepts and facilities of this product are described in such a way that developers and BI specialists get a feeling of how this product works, what its features are, and what it would mean to develop a DDP-based business intelligence architecture using this product.*

The Data Delivery Platform is a business intelligence architecture that offers many advantages for developing BI systems, including increased flexibility of the architecture, shareable transformation and reporting specifications, easy migration to other data store technologies, cost reduction due to simplification of the architecture, easy adoption of new technology, and transparent archiving of data. The DDP can co-exist with other more well-known architectures, such as the Data Warehouse Bus Architecture, and the Corporate Information Factory.

Informatica Data Services (Informatica Data Services) is a data integration platform. Basically, it's a *data federation* and *virtualization server* extended with more classic ETL (Extract Transform Load) functionality. Informatica Data Services can present a heterogeneous set of data stores as one logical data store. This unified view can be used by almost any reporting and analytical tool, and in addition it can be accessed by applications through service-oriented interfaces. Informatica Data Services offers the following features:

- On-demand (i.e. real-time or federated) and scheduled (i.e. batch) data integration capabilities in a single environment
- Integrated collaborative data profiling and on-demand data cleansing features
- Wide range of transformation operations, including complex transformations and cleansing operations
- Shareable transformation specifications
- Integrated lineage and impact analysis
- Advanced security rules for data access
- Advanced query optimization techniques
- Sophisticated caching mechanism
- Access of any type of data source (e.g. structured, unstructured, semi-structured, cloud, archived)
- Easier data store migration
- Integration of cloud data sources
- Transparent archiving of data and federation of production and archived data

Pure ETL tools integrate data using *scheduled transformation* while traditional data federation servers limit users to transformations offered by SQL or XQuery. Scheduled transformation means that the data stores accessed by the reporting tools are refreshed periodically. It also means that several derived data stores must be developed and maintained to store the periodically copied data. With *on-demand transformation*, data is retrieved from the data stores the moment reporting tools request data, and only then is the data transformed. The advantages of scheduled transformation are that users can work with more timely data, less need exists for creating and managing derived data stores, and report and transformation changes can be applied more quickly. Informatica Data Services supports both scheduled and on-demand transformations including mid-stream data profiling of federated data and application of data quality rules in real-time on the federated data. These features make Informatica Data Services ideal for developing a Data Delivery Platform.

To summarize, Informatica Data Services is an advanced, mature, and feature-rich open data federation and data virtualization server. It elegantly combines typical federation or data virtualization features with those of an ETL solution. This makes the product suitable for scheduled and on-demand transformations. Informatica Data Services modular approach, flexibility, support for standards, and extensive optimization technologies make it very well suited for developing a business intelligence system based on the Data Delivery Platform architecture.

2 Business Intelligence Trends

The reporting and analytical needs of managers and decision makers have changed over time. In the beginning, users were satisfied if they could run simple reports that would, for example, show them the total amount of sales per region. In addition, the list of reports they could run was normally pre-defined; they couldn't develop new reports themselves. That was done by reporting specialists in the IT department.

It didn't take very long before users requested more advanced analytical and ad hoc capabilities that would allow them to create new reports themselves. To address user needs, vendors released so-called *managed query tools* followed by *OLAP tools* (OnLine Analytical Processing). Both classes of tools gave users more dynamic query capabilities, such as drill-downs, rollups, and the ability to create new reports themselves. They were able to do all this without having to understand SQL or database technology. But the need for more analytical capabilities didn't stop here. Users continued to increase their demands and needs. Some of those new reporting and analytical trends are:

Operational Reporting and Analytics – 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 good enough for most users. Nowadays, more and more users are demanding access to data that is (almost) 100% up-to-date. In other words, these users want to work with (near) real-time data. This new form of reporting and analytics is usually referred to as o*perational reporting and analytics*.

There are many examples of environments that need operational reporting and analytics. For example, a retail company might 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 highly 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.

Deep Reporting and Analytics – For many reports and forms of analytics, storing detailed data is not necessary; aggregated data or slightly aggregated data is sufficient. For example, to determine the total sales per region, no need exists to store and analyze all the individual sales records. Aggregating the data on, for example, customer number is probably adequate. But for some forms of analytics, detailed data is needed. This is called *deep analytics* or *big data analytics*. If an organization wants to analyze whether trucks should be rerouted, or if it wants to determine which online ad to present, detailed data must be analyzed. And the most well-known area that requires detailed data is time-series analytics. The consequence of analyzing detailed data is that the data stores will grow enormously, potentially leading to serious problems with query performance.

Self-Service Reporting and Analytics – Before users can run their reports, the IT department must set up an entire environment, which takes some time. *Self-service reporting and analytics* implies that users can develop their own reports with a minimal setup required. Self-service reporting is useful when a report must be developed quickly and there is no time to prepare a complete environment. For example, an airline wants to know how sales will be affected by a particular strike tomorrow. Another example is when a requested report will be used only once. In that case, self-service analytics can be very helpful. For both examples, it would not make sense to first develop a dedicated data store and ETL scripts to fill the data mart before running the reports. For the first example, creating the derived data store would take too long, and for the second example, it's not worth the effort.

Complex Reporting and Analytics – The complexity of user demands keeps increasing. Besides standard reports, users want to create and run complex statistical models. They may want to create forecasting models (i.e. a retailer might want to see the impact of a price increase on expected sales), predictive models (i.e. an insurance company might want to predict which customers will be more interested in particular insurance combinations), and optimization models (i.e. a transportation company wants to know what the most efficient route is for a truck to deliver goods to various stores). Some of these are pure data mining algorithms requiring complex to very complex queries. And for some of them, large portions of the data warehouse must be scanned.

To summarize, these new forms of reporting and analytics require that more data, and also more detailed data, must be stored, access to more up-to-date is required, and the infrastructure should be more flexible. These new demands will have a serious impact on the supporting business intelligence architecture. This architecture normally consists of data stores, such as data warehouses, operational data stores, and data marts, and of ETL logic to transform, cleans, and copy data between data stores. With most of the current business intelligence architectures, implementing these new demands will be hard. New architectures are needed. One of those new architectures is the *Data Delivery Platform* which is described in the next section.

3 What is the Data Delivery Platform?

The *Data Delivery Platform* (DDP) is a flexible architecture for developing business intelligence systems where data consumers (such as reports developed with SAP BusinessObjects WebIntelligence, SAS Analytics, JasperReport, and Excel) are decoupled from data stores (such as data warehouses, data marts, and staging areas); see Figure 1.

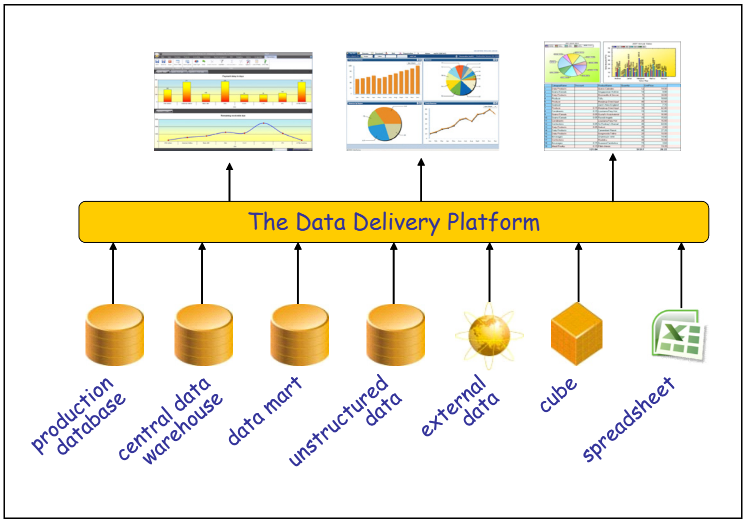


Figure 1 *The Data Delivery Platform*

In a more classic business intelligence architecture reporting tools normally access data stores directly. In a way, they are tied to those data stores. In the DDP they are not accessing data stores but an intermediate layer of software. This layer makes sure that requests from the data consumers are directed to the right data store(s). Because of the intermediate layer, the data consumers don’t know (nor do they have to) from which data store(s) the data is coming from. It could be that a particular report is receiving the requested data from the central data warehouse, the other from a combination of a data mart and a spreadsheet, and the third from the combination of a production database and two cubes.

The primary goal of decoupling is to get a higher level of flexibility. For example, changes made to the data stores don’t automatically imply that changes must be made to the data consumers as well, and vice versa. Or, replacing one data store technology by another is easier when that data store is ‘hidden’ behind the DDP.

The DDP was introduced in a number of articles published at BeyeNETWORK.com, including

The Definition of the Data Delivery Platform[1]. The definition of the DDP is:

> *The Data Delivery Platform is a business intelligence architecture that delivers data and meta data to data consumers in support of decision-making, reporting, and data retrieval; whereby data and meta data stores are decoupled from the data consumers through a meta data driven layer to increase flexibility; and whereby data and meta data are presented in a subject-oriented, integrated, time-variant, and reproducible style.*

Decoupling data consumers from data stores is based on the concept of *information hiding*. This concept was introduced by David L. Parnas[2] in the 70s and was adopted soon after by object oriented programming languages, component based development, and service oriented architectures. The concept of information hiding is slowly receiving more interest in the world of data warehousing. However, the terms currently used to refer to this concept are data abstraction and data virtualization.

The DDP can be seen as a separate business intelligence architecture, but it can also co-exist with more well-known architectures, such as Ralph Kimball's *Data Warehouse Bus Architecture*[3], Bill Inmon's *Corporate Information Factory*[4], and his more recent architecture called *Data Warehouse 2.0*[5]*.* In addition, some other generic architectures exist, such as the *Centralized Data Warehouse Architecture* and the *Federated Architecture*; see the article *Which Data Warehouse Architecture is Most Successful?* by T. Ariyachandra and H.J. Watson, published in 2006[6].

We would like to emphasize that it's not the intention of the DDP to replace the data warehouse concept, but to complement it. In most organizations, production systems have been developed in such a way that a data warehouse will always be needed for reporting. One reason might be that if a production system doesn't keep track of historical data, that data has to be kept somewhere else, for example in a data warehouse. Another reason might be that the current workload on the production systems is so intense, that reporting directly on the production databases, might lead to too much interference. Again, a data warehouse might be the solution.

[1] See http://www.b-eye-network.com/channels/5087/view/12495.

[2] David L. Parnas, *'Software Fundamentals, Collected Papers by David L. Parnas'*, Addison-Wesley Professional, 2001.

[3] R. Kimball et al., *The Data Warehouse Lifecycle Toolkit*, Second Edition, John Wiley and Sons, Inc. 2008.

[4] W.H. Inmon, C. Imhoff, and R. Sousa, *Corporate Information Factory*, Second Edition, John Wiley and Sons, Inc., 2001.

[5] W.H. Inmon, D. Strauss, and G. Neushloss, *DW 2.0, The Architecture for the Next Generation of Data Warehousing*, Morgan Kaufmann Publishers, 2008.

[6] See http://www.tdwi.org/Publications/BIJournal/display.aspx?ID=7890.

4 Advantages of the Data Delivery Platform

As indicated, the *Data Delivery Platform* (DDP) is a business intelligence architecture. Two principles are very fundamental to this architecture: *shared specifications* and *decoupling* of data consumers and data stores. Both these principles lead to a number of advantages that are described in this section.

Most reporting and analytical tools require *specifications* to be entered before reports can be developed. Some of those specifications are *descriptive* and others are *transformative*. Examples of descriptive specifications are definitions of concepts; for example, a customer is someone who has bought at least one product, and the Northern region doesn't include the state Washington. But defining alternative names for tables and columns, and defining relationships between tables are also descriptive specifications. Examples of transformative specifications are 'how should country codes be replaced by country names', and 'how should a set of tables be transformed to one cube'. In the DDP those specifications are centrally managed and are shareable. The advantages resulting from shared specifications are:

Easier Maintenance of Specifications – Unfortunately, in most cases, descriptive and transformative specifications can't be shared amongst reporting and analytical tools. So, if two users use different tools the specifications must be copied. The advantage of the DDP is that most of those specifications can be defined once and can be used by all the tools. Therefore, maintaining existing and adding new specifications is easier, which makes self-service reporting and analytics easier to implement.

More Consistent Reporting – If all reporting and analytical tools use the same specifications to determine results, the results will be consistent, even if the tools are from different vendors. This improves the perceived quality of and trust in the business intelligence environment.

Increased Speed of Report Development – Because most specifications already exist within the DDP and can be re-used, it takes less time to develop a new report. Development can focus primarily on the use of the specifications.

In a DDP data consumers are *decoupled* from the data stores. This means that the data consumers don't know which data stores are being accessed: a data warehouse, a data mart, or an operational data store. Neither do they know which data store technologies are being accessed, an Oracle or DB2 database or maybe Microsoft Analysis Service. The advantages resulting from this decoupling are:

Easier Data Store Migration – The DDP offers data store independency which means that if a report accesses a particular data store it can easily be migrated to another. The reports' queries can be redirected through the DDP to that other data store. For example, if a report is currently accessing a data mart, migrating it to the data warehouse doesn't require any changes in the report definition. The same applies if a need exists to migrate from a relational database to

MDX-base technology, or if SQL Server has been replaced by Netezza. In most cases, these changes will have no impact on the reports. In short, if a DDP is in place, migration to another data store (technology) is easy. There are various reasons why an organization wants to migrate, for example, they may want to use technology that offers faster query performance, or data storage is outsourced and needs to be accessed differently.

Cost Reduction Due to Simplification – If the DDP is installed in an existing business intelligence architecture, for example, one based on the Corporate Information Factory architecture, the DDP makes it possible to simplify the architecture. Data marts and cubes can be removed and the existing reports must be redirected to another data store, which, as indicated, is easy to do with the DDP. The advantage of this simplification of the architecture is cost reduction.

Increased Flexibility of the Architecture – With less code and fewer specifications, it's easier to change a system. The DDP makes it possible to simplify the architecture and to work with shareable specifications. The effect is that new user requirements and demands can be implemented faster. In other words, the time to market for new reports is significantly shortened. This fits the need for more self-service reporting and analytics.

Access to Operational Data – If production databases have also been hooked-up to the DDP, data consumers can be given access to operational data. This allows for joining historical data coming from the central data warehouse with data stored in production databases. The advantage is that the requirements of those users who need operational reporting and analytics can be met. This approach does require that care should be taken with a possible interference on the production system.

Seamless Adoption of New Technology – Recently, powerful new database and storage technology has been introduced, such as data warehouse appliances, analytical databases, columnar databases, and solid state disk technology. As indicated, because the DDP separates data consumers from data stores, replacing an existing data store technology with a new one is relatively easy and has no impact on the reports.

Transparent Archiving of Data – Eventually, data warehouses might become so massive that 'older' data has to be archived. Normally, this means that data is taken from the original data store and moved to another. But if data is old, it doesn't always mean users don't want to access it anymore. The consequence might be that certain reports have to be rewritten to access the data store that contains the archived data. If a DDP is in place, it can hide where and how archived data is stored. If users are still interested in all the data, the DDP can combine the non-archived data with the archived data store in real-time. Depending on the implementation, the effect might be that the performance is slower, but reports don't have to be changed. Again, the DDP transparently hides that some data has been archived.

To summarize, the DDP is a modern and flexible business intelligence architecture designed for classic and the latest reporting and analytical requirements. But how can a Data Delivery Platform be developed? Currently, the most practical approach is to use a *federation server*. This is the topic of the next section.

5 Data Virtualization, Data Federation, and Data Integration

A federation server is a product that offers data federation capabilities. This section explains what data federation is, but it begins with explaining what the term data virtualization means. Although the two have a strong relationship, they are different. The end of this section focuses on a third related term: data integration.

Data Virtualization – *Virtualization* is not a new concept in the IT industry. It all started years ago when virtual memory was introduced in the 1960s using a technique called paging. Memory virtualization was used to simulate more memory than was physically available in a machine. Nowadays, almost everything can be virtualized, including processors, storage, networks, and operating systems. In general, virtualization means that applications can use a resource without concern for where it resides, what the technical interface is, how it has been implemented, which platform it uses, how large it is, and how much of it is available.

The definition of data virtualization:

> *Data virtualization is the process of offering data consumers a data access interface that hides the technical aspects of stored data, such as location, storage structure, API, access language, and storage technology.*

Data virtualization provides an abstraction layer that data consumers can use to access data in a consistent manner. A data consumer can be any application retrieving or manipulating data, such as a reporting or data entry application. This abstraction layer hides all the technical aspects of data storage. The applications don’t need to know where all the data has been stored physically, where the database servers run, what the source API and database language is, and so on.

Technically, data virtualization can be implemented in many different ways. Here are a few examples:

- With a federation server multiple data stores can be made to look as one. The applications will see one large data store, while in fact the data is stored in several data stores. More on data federation next.
- An Enterprise Service Bus (ESB) can be used to develop a layer of services that allow access to data. The applications invoking those services don’t know where the data is stored, what the original source interface is, how the data is stored, and what its storage structure is. They will only see, for example, a SOAP interface. In this case, the ESB is the abstraction layer.
- Placing data stores in the cloud is also a form of data virtualization. To access a data store, the applications will see the cloud API, but they have no idea where the data itself resides. Whether the data is stored and managed locally or whether it’s stored and managed remotely is transparent.

- In a way, building up a virtual database in memory with data loaded from data stored in physical databases can also be regarded as data virtualization. The storage structure, API, and location of the real data is transparent to the application accessing the in-memory database. In the BI industry this is now referred to as *in-memory analytics*.
- Organizations could also develop their own software-based abstraction layer that hides where and how the data is stored.

Data Federation – As indicated, one way to implement data virtualization is by using *data federation*. In other words, data federation is a form of data virtualization. Here is the definition of data federation:

> *Data federation is a form of data virtualization where the data stored in a heterogeneous set of autonomous data stores is made accessible to data consumers as one integrated data store by using on-demand data integration.*

This definition is based on the following important concepts:

- Data virtualization: Data federation is a form of data virtualization. Note that not all forms of data virtualization imply data federation. For example, if an organization wants to virtualize the database of one application, no need exists for data federation. But data federation always results in data virtualization.

- Heterogeneous set of data stores: Data federation should make it possible to bring data together from data stores using different storage structures, different access languages, and different API's. An application using data federation should be able to access different types of database servers and files with various formats; it should be able to integrate data from all those data sources; it should offer features for transforming the data; and it should allow the applications and tools to access the data through various APIs and languages.

- Autonomous data stores: Data stores accessed by data federation are able to operate independently, in other words, they can be used outside the scope of data federation.

- One integrated data store: Regardless of how and where data is stored, it should be presented as one integrated data set. This implies that data federation involves transformation, cleansing, and possibly even enrichment of data.

- On-demand integration: This refers to *when* the data from a heterogeneous set of data stores is integrated. With data federation, integration takes place on the fly, and not in batch. When the data consumers ask for data, only then data is accessed and integrated. So the data is not stored in an integrated way, but remains in its original location and format.

From this definition can be derived that a *data federation server* or, for short, a *federation server*, is a product that implements data federation. All this implies that a federation server should be able to access different types of database servers and files with various formats; it should be able to integrate data from all those data sources; it should offer features for transforming the data; and, it should allow the applications and tools to access the data through various APIs and languages.

For those applications accessing a federation server it's very similar to logging on to a particular database. Without knowing it, reports join data coming from different data stores, even data stores that use different storage models and concepts. Because a federation server presents this unified view of the data, it can act as the core building block of a DDP. A federation server will take care of all the integration and transformation required to join data from different data stores together.

Note that not all forms of data virtualization imply data federation. For example, if an organization wants to virtualize the database of one application, data federation is not a necessity. But data federation always leads to data virtualization.

Data Integration – According to SearchCRM, the term integration (from the Latin word *integer*, meaning whole or entire) generally means combining parts so that they work together or form a whole. If data from different data sources is brought together we talk about *data integration*:

> *Data integration is the process of combining data from a heterogeneous set of data stores to create one unified view of all that data.*

Data integration involves joining data, transforming data values, enriching data, and cleansing data values. What this definition of data integration doesn't enforce is how integration takes place. For example, it could be that original data is copied from its source data stores, transformed and cleansed, and subsequently stored in another data store. This is the approach taken when using ETL tools. Another solution would be if the integration takes place live. For example, a federation server would do most of the integration work on-demand. Another approach is that the source data stores are modified in such a way that data is transformed and cleansed. It's like changing the sources themselves in such a way that almost no transformation and cleansing is required when data is brought together.

To summarize, data virtualization might not need data integration. It depends on the number of data sources being accessed. Data federation always requires data integration. For data integration, data federation is just one style of integrating data.

6 Open versus Closed Federation Servers

Various analytical and reporting tools implement a form of federation technology themselves. For example, an analytical tool such as Qlikview is more than capable of accessing a set of heterogeneous data stores, the same applies for SAP/Business Objects, IBM/Cognos, and many others. For example, the Universe concept in Business Objects can be seen as federation technology. However, all the specifications entered in these products are only usable by the tools themselves (or tools of the same vendor); see Figure 2. These are non-sharable specifications.

Therefore, these federation technologies are called *closed federation servers*.

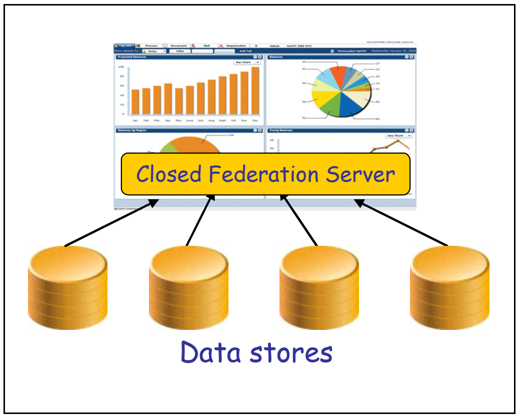


Figure 2 *A closed federation server working for only one reporting tool*

Like a closed federation server, an *open federation server* can access many different data stores, but in addition it also allows access for any BI tool and application; see Figure 3. The effect is that specifications stored in the federation server become shareable. If, for example, we define that the Northern Region doesn't include the state Washington, each and every tool that accesses the federation server can make use of that same specification, whether it's Excel or SAS Analytics. This improves the maintainability of the environment, but it also minimizes the chance that users using different tools see inconsistent results.

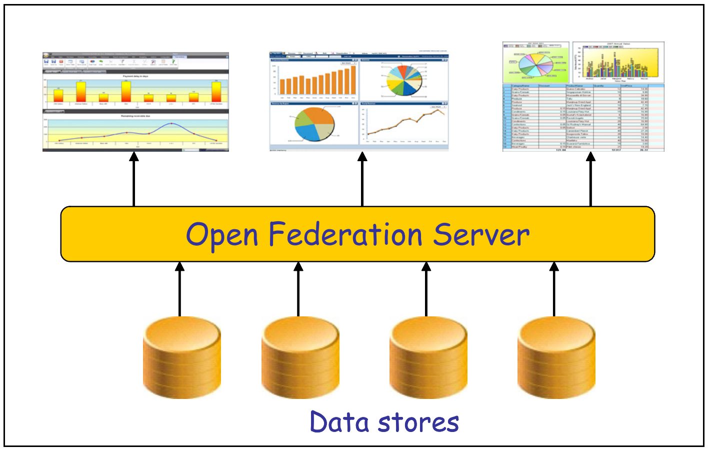


Figure 3 *An open federation server with shareable specifications*

7 Federation Servers offer On-Demand Transformation

Scheduled Transformations with ETL – In most current business intelligence architectures the format and the contents of the data stored in the source systems are quite different from how the users want to see the data in their reporting and analytical tools. For example, in the source systems customer data might be spread out over multiple databases, while users want to have an integrated view; or, data in the source systems might be heavily coded, while users want to see meaningful values; or, historical data might be missing from the source systems, while users need it for trend analysis; or, the values of data elements in source systems might be incorrect (defective data), while users want to work with correct data, and so on. To summarize, source data has to be 'massaged' before users can use it. This whole process of massaging is sometimes referred to as *data transformation.*

In most current business intelligence systems data transformation is handled by a solution that copies the relevant data from one data store to another. And somewhere during this copy process data is transformed to the right format. In addition, all the data transformation is often not done in one step, but in multiple. For example, data is first copied from an operational database to an operational data store, onwards to the data warehouse, and finally it's copied to a data mart. With each step the format of the data gets more and more like what the reporting tools need. A consequence of this approach is that several data stores with redundant data must be kept and maintained; see Figure 4.

Normally, all this copying and transforming of data is handled by an *ETL tool* (Extract Transform Load), such as Informatica's PowerCenter. ETL jobs are developed and run to take care of this. ETL jobs are scheduled to run periodically, for example, once a week, every midnight, or twice a day.

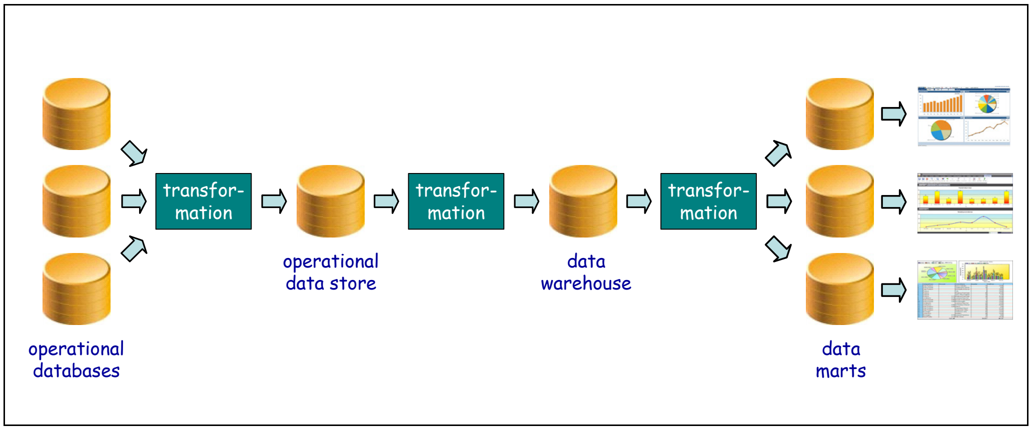


Figure 4 *Scheduled transformation requires data stores to hold intermediate and final results*

This form of transformation, where data is physically copied after every transformation step and where data is periodically refreshed, is called *scheduled transformation* in this whitepaper. The advantages of scheduled transformation are that data is copied and transformed only a limited number of times, and that the data in the data stores accessed by the applications is rather static and can therefore easily return consistent reporting results. However, disadvantages are that the reporting tools have no access to current data, a lot of redundant data is stored, and a complex architecture had to be developed.

On-Demand Transformations with Federation Servers – Federation servers, such as Informatica Data Services, offer another form a transformation, called *on-demand transformation*. When a user executes his report or runs his analysis, only then is data queried and integrated. It's almost as when the report is executed, the data flows from the databases through the federation server to the reports. The federation server will do the necessary transformations. Less need exists for developing derived data stores, such as data marts. Therefore, reports using a federation server can have access to current data; see Figure 5.

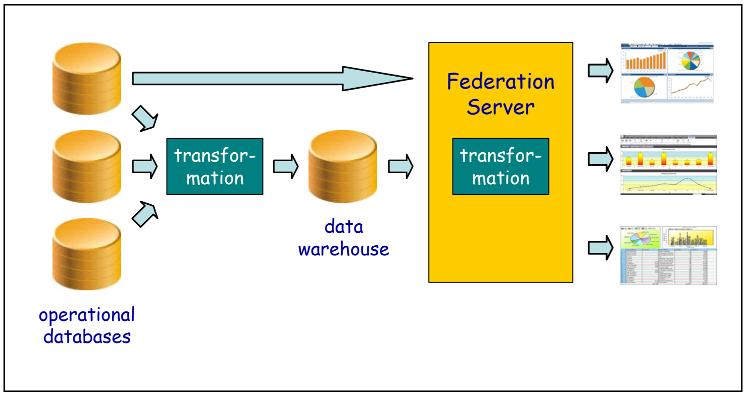


Figure 5 *On-demand transformation requires less data stores*

The essential difference between on-demand and scheduled transformation can be explained using the following analogy. On-demand transformation (working with a federation server) can be compared to buying a sandwich that is prepared right in front of you, versus one that has been prepared in a factory, wrapped in plastic, and sent to a store (scheduled transformation).

On-demand transformation offers users the following advantages:

- Users can work with more timely data.
- Less need exists for creating derived data stores that will reduce costs.
- With less data stores, the overall architecture will be more lightweight, and that means the whole architecture is more flexible.
- The time-to-market for new reports is better; creating a new report might only require a few hours work before the required data is available.
- Sometimes reports must be developed that will run only once, on occasion these are called throw-away reports. If a report is used only once, on-demand transformation fits better.

But some disadvantages of on-demand transformation must be considered as well:

- Similar transformations are executed repeatedly. Every time a report accesses data, that data will go through the same transformations (unless a cache is used, see Section 23). Compare this to scheduled transformation, where data is transformed only once or a few times, in fact only when data is added or changed. Note that this is only true for those solutions where only new data is copied; in some organizations all the data is copied every time. In that case, the same transformations are also done repeatedly.
- The transformation might be so complex that it takes too long.
- Some production systems overwrite old data when new data is entered. If this historical data is needed, it must be copied from the source for retention in a special data store (for example a staging area or operational data store) using a scheduled transformation approach.

Whether on-demand or scheduled transformation is applied, each tool involved in data transformation should support operations such as:

- Transformation: Values in columns are changed by applying, for example, concatenations, code transformations, calculations, and string manipulations.
- Join: Data in two or more tables are combined into one table.
- Aggregation: Rows are grouped based on equal values in certain columns.
- Selection: Rows are selected based on specified conditions.
- Projection: Columns are selected, and others are removed.

One issue that hasn't be mentioned yet is what happens if the data cannot or should not be transformed because the data is not according to the rules that apply to the data. In other words, how is defective data handled?

8 On-Demand Transformation and Cleansing Data

Through the years, various studies have shown that handling *defective data* (or dirty data) is still one of the biggest challenges when designing and implementing a business intelligence architecture. To improve the quality of the reports and analysis, most organizations have to cleanse defective data before it is used. Several dedicated cleansing tools exist on the market to help out with this. These tools can, for example, help with the spelling of company names, they can find the correct zip code belonging to an address, and they can de-duplicate records.

With scheduled transformation, cleansing operations can be applied during the processing of ETL jobs. When an ETL job runs, the cleansing operations are regarded as just one of the many transformation operations, just like join and filter. As an example, Figure 6 contains a screenshot of a PowerCenter ETL script that invokes such as cleansing algorithm. Some cleansing operations are very complex and can be quite time-consuming. Because all operations in an ETL environment are run in the batch, there is often sufficient time available.

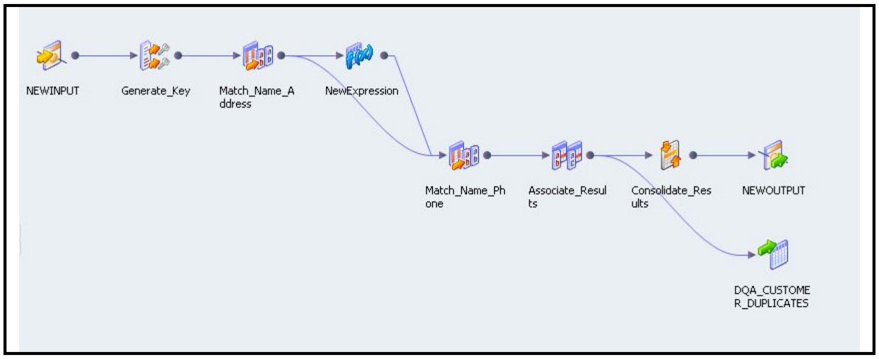


Figure 6 *An example of a PowerCenter ETL script where cleansing operations are executed*

If on-demand transformation is used, cleansing should be performed as part of the query processing. The consequence is that cleansing might not take too much time. Some of the simple forms of cleansing can easily be implemented in an on-demand transformation solution, because they won't consume too much time. For example, if the values of a column called GENDER can only be equal to `M` and `F`, and if a record exists containing the value `m`, this type of cleansing is simple. It's a simple string manipulation. Or if a zip code contains a blank where it shouldn't, that is another form of cleansing that's easy to fix. Applying one or two string manipulations functions will solve the problem. But beyond these simple cases, it's not always recommended to perform the more complex cleansing operations using on-demand transformation. Else, the cleansing would slow down the transformation process too much. A different solution has to be found.

Informatica Data Services does allow that cleansing operations are activated in an on-demand style. In this respect, it's a rather unique product. However, if particular cleansing operations take too long, a partly scheduled and partly on-demand solutions can be selected.

9 Application Areas of Federation Servers

As explained in the previous sections, with a federation server a set of data stores can be made to look like one integrated data store. Here are some of the application areas of this technology:

Virtual Data Mart – A federation server can be used to simulate the existence of a virtual data mart that presents data stored in a data warehouse and that uses on-demand transformations. Instead of having to physically create a data store plus ETL scripts for loading the data mart, only the required table structures in the data mart must be defined.

Virtual Data Warehouse – If a business intelligence architecture consists of data marts only and no data warehouse, and if some users want to run reports using data from multiple data marts, a federation server can help out. A virtual data warehouse can be created that uses data stored in the physical data marts. The federation server will join tables from multiple data marts together the moment a report asks for the data. To the report it will look as if the accessed tables come from one integrated data store. Note this will only work if the dimensional tables in the merged data marts are conformed, meaning, only if they can be integrated.

Virtual Central Data Warehouse – In some organizations multiple, independent BI architectures are developed, each with its own central data warehouse. If reporting of data stored in all of these central warehouses is required, a virtual data warehouse can be developed using a federation server.

Extended Data Warehouse – Reports that need to join data from the data warehouse with data from other data sources, such as operational databases, external sources and local files, a federation server could present an integrated view of all those data stores. In a way, the federation server extends the data warehouse with other data stores without having to copy data from those other data stores to the warehouse.

Operational Data Warehouse – By giving a federation server access to a data warehouse plus to some operational databases, reports can join historical data from the data warehouse with 100% up-to-date data from operational databases, thereby simulating an operational data warehouse (sometimes referred to as an online or near-online data warehouse).

***Virtual Sandboxing* –** Sandboxing involves the creation of an environment where power users can explore data and find answers to unanticipated questions. With a federation server a virtual sandbox be created. This solution involves less work and less investment than when a physical sandbox is developed.

***Prototyping* –** Before a complex solution is developed using data marts and scheduled transformation, it might be useful to first develop a solution without the need of extra data stores that uses on-demand transformation. This prototype might show, for example, which transformation and cleansing problems to be tackled, and it might show users what the reports will look like. Afterwards, the real version is developed using all the knowledge picked up doing the prototype.

***Throw Away Reports* –** Occasionally, particular reports must be developed urgently. And in most cases those reports will be run only once. Due to the urgency, there is no time to set up a physical environment consisting of, for example, a data mart and ETL scripts. With a federation server, this environment, probably consisting of a number of virtual tables, can be developed easily. After the report has been run, the virtual tables can be deleted.

***Self-Service Reporting and Analytics* –** Self-service reporting and analytics means letting the users themselves develop their reports and do their own analytics. The IT department remains responsible for setting up the data stores for them. With a federation server these stores can be setup very quickly and can be adapted easily, because they are virtual and not physical.

***Enterprise Data Sharing* –** Operational applications might need to access multiple heterogeneous operational databases. A federation server might be able to hide all the technical and semantical differences between the various systems, and present one logical view to these operational applications.

***Data Services* –** Most federation servers are able to offer SOAP-based service access to the data stored in databases. In other words, with federation servers service interfaces for querying and manipulating stored data can be developed quite easily. This is very convenient if a service oriented architecture is being developed.

***Data Mashups* –** For mashups with a focus on querying and manipulating data from different data stores, a federation server can speed up their development. The federation server will handle access to all the data stores, will handle the necessary integration, cleansing, and transformation of data, whereas the mashup can focus on the user interface aspects of the application.

10 What is Informatica Data Services?

Several federation servers are available on the market today. This whitepaper describes how to develop a DDP with the open federation server called *Informatica Data Services* from Informatica Corporation.

Informatica Corporation – Redwood City, California based Informatica Corporation was founded in 1993. Their first product was PowerCenter, a high-end ETL tool. From the beginning, PowerCenter has always been a dominant and popular ETL product for developing business intelligence systems. For years, Gartner positioned Informatica as a leader in the top right-hand quadrant of their data integration quadrant. Currently, Informatica reports having a customer base of over 2700 companies.

Informatica Data Services – A few years ago, Informatica realized that the reporting and analytics needs of users where shifting. The need for on-demand integration would increase accordingly. In 2007 they decided to develop a separate product for data integration which became Informatica Data Services.

After three years of research and development, the first version of Informatica Data Services was released in December, 2009. Informatica Data Services combines on-demand integration with scheduled transformation. Besides the standard features one expects from every federation server, Informatica Data Services supports some unique features, such as on-demand transformation as well as scheduled transformation, and data profiling logic is an integral part of the product.

Informatica Data Services has been developed on the Informatica 9 platform, the same platform on which PowerCenter has been built. Therefore, the first and current version of Informatica Data Services has number 9.0.1. In the first half of 2011 version 9.1 will be released.

Besides being able to access almost any relational database server, including DB2, Microsoft SQL Server, MySQL, Netezza, and Oracle, Informatica Data Services makes it possible to access XML documents, flat files, spreadsheets, and other non-relational data stores. Informatica Data Services will 'flatten' non-relational stores to tables. This makes it possible that, for example, data in an Excel spreadsheet is joined with data stored in an Oracle database, and data inside an XML document.

Reporting tools can use any of the popular API's to access data, including JDBC and ODBC. In addition, Informatica Data Services can present data as services through SOAP over HTTP/S, and applications can access data through JMS as well.

Because Informatica Data Services can create a service layer on top of a set of data stores, it can operate perfectly within a SOA architecture. These services can be used as so-called *data services*, and will be responsible for retrieving and manipulating data. These services will integrate and cleanse the data. Using Informatica Data Services in this way simplifies development of an SOA. Services at the lowest level can be developed easily. In fact, investments that have already been made for the business intelligence architecture can be re-used. The two architectures will also become more aligned, because they share the same logic for integration, transformations, and cleansing.

Informatica Data Services and PowerCenter – Although PowerCenter and Informatica Data Services are both data integration products, they are not competing products. PowerCenter has always been positioned for those environments that need high-end scheduled transformations, requiring large sets of data to be moved and transformed in a fast and scalable fashion. Informatica Data Services, on the other hand, is positioned as complementary for those environments that require on-demand integration plus some scheduled transformation. The main difference between the products is the way they have been optimized internally, one for on-demand and the other for scheduled transformations.

The two products can co-exist seamlessly. Specifications stored in the repository of one can be copied to the other. Informatica has also reused proven optimizations and data processing algorithms from PowerCenter in Informatica Data Services to increase robustness of the product. For example, all internal code for data transformations and data processing is common code across the products.

The next sections in this whitepaper show how developers work with Informatica Data Services and describe the features of the product.

11 Informatica Data Services Under the Hood

The next sections describe how particular features and concepts are implemented in Informatica Data Services. This should give readers a better understanding of how a BI system with an architecture based on the Data Delivery Platform can be developed. The following features are described:

- Defining virtual tables, physical data objects, and mappings.
- Sharing specifications by stacking virtual tables.
- Integrated and on-demand data profiling.
- Collaborative and on-demand data cleansing.
- Developing virtual data marts.
- Making XML documents and spreadsheets accessible as tables.
- Tracking all the relationships between virtual tables and other objects (lineage analysis).
- Exposing tables as data services.
- Caching data for performance reasons.
- Optimization techniques for running queries.
- Securing access to data.

12 Defining Virtual Tables

Data made available through Informatica Data Services is organized in tables consisting of rows and columns. Informatica Data Services offers various APIs for accessing those tables. One application can access the table through, for example, a classic JDBC/SQL interface and the other can access that same table through a SOAP-based interface. Both applications will see the same data. But neither has any idea where or how the data in those tables is stored; see also Figure 7.

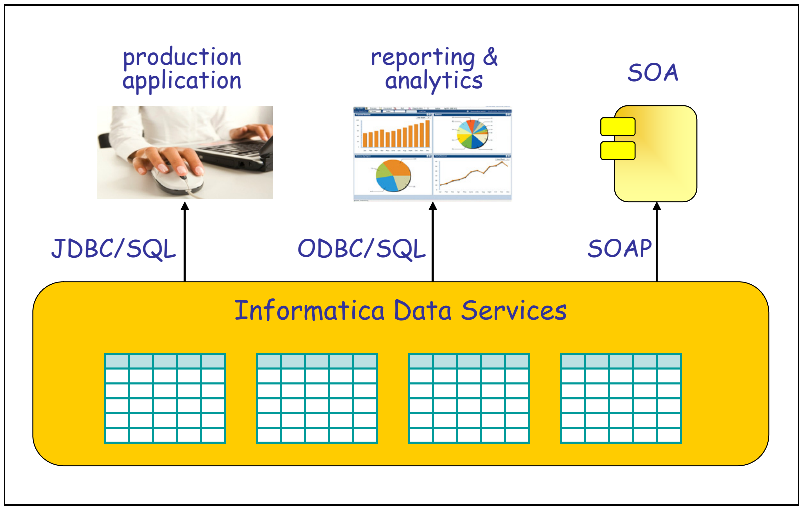


Figure 7 *Applications can access the data in Informatica Data Services through different interfaces*

From Virtual Tables to Foreign Tables – Accessing an Informatica Data Services table is very similar to accessing a table managed by a classic SQL database server, such as Oracle, MySQL, or SQL Server. However, an Informatica Data Services table doesn't store data and doesn't occupy storage, its contents is virtual, which is why it's called a *virtual table* in Informatica Data Services. When a virtual table is accessed, Informatica Data Services retrieves the real data from some data source(s). A data source doesn't have to be a table stored in a SQL database, but can also be, for example, a flat file, a spreadsheet, or an XML document. These data sources are called *foreign tables* in this whitepaper, because they are defined and managed outside Informatica Data Services. This name also makes it easier to distinguish them from virtual tables.

The definition of a virtual table contains descriptions of all its columns, their respective data types, and relationships with other virtual tables. Together this forms the structure of the virtual table.

Before a virtual table can be used, it has to be linked to one or more foreign tables. This is done in two steps. In the first step, by creating a *physical data object* for a foreign table, the latter is introduced in Informatica Data Services. The definition of a physical data object contains the technical structures of the real data source plus specifications that deal with where and how the real data source is stored and how to access it. Most of the specifications are very technical by nature. In a way, a PDO is a wrapper around a foreign table.

In the second step, the virtual table is linked to one or more physical data objects. This is done through a so-called *data object mapping*. Such a mapping defines how the data from those underlying physical data objects should be mapped and transformed to become the virtual contents of a virtual table. As long as a virtual table is not linked, it can't be queried by applications.

In a way, a virtual table is very much comparable to the view concept in SQL database servers. Basically, a view is a query definition with a name and a set of columns. The contents of the view is virtual and is derived from underlying tables and views. A virtual table in Informatica Data Services has a query definition as well, although not specified in SQL but with a data object mapping, and the contents of a virtual table is also virtual and is derived from underlying data sources, the physical data objects. Later in this whitepaper we will discuss the notion of a cached virtual table, meaning its contents is not virtual, but is stored (temporarily) in a cache.

Figure 8 visualizes the relationships between the concepts virtual table, data object mapping, and physical data object.

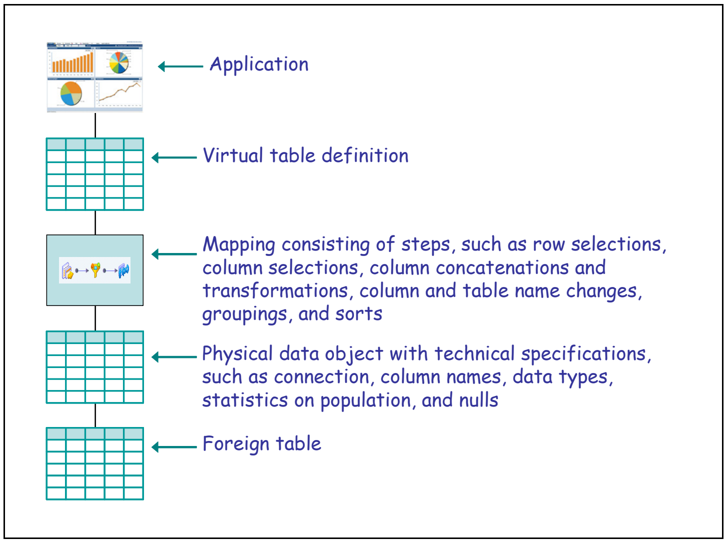
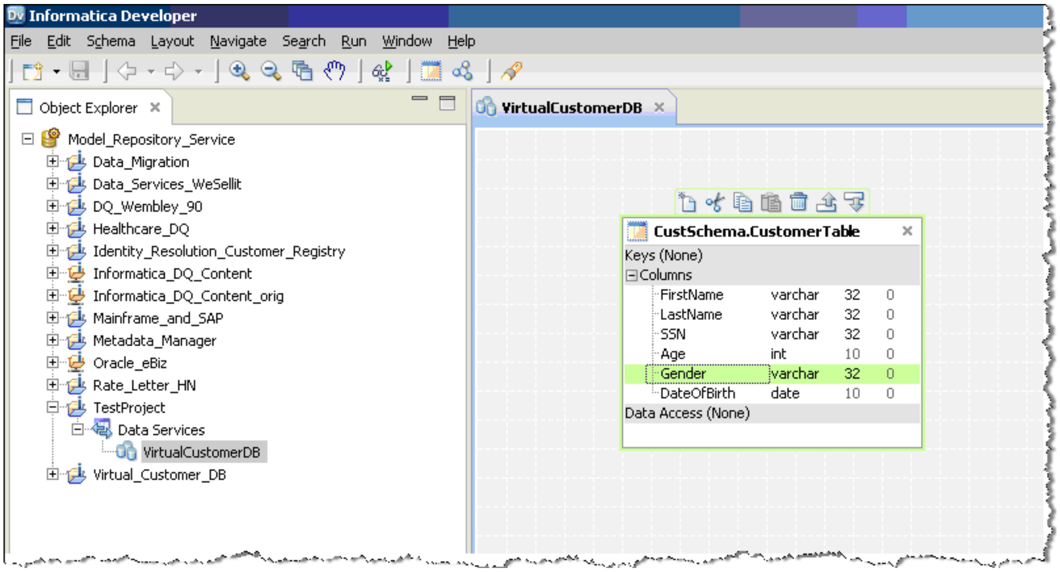

Application
Virtual table definition
Mapping consisting of steps, such as row selections, column selections, column concatenations and transformations, column and table name changes, groupings, and sorts
Physical data object with technical specifications, such as connection, column names, data types, statistics on population, and nulls
Foreign table

Figure 8 *The relationships between virtual table, data object mapping, physical data object, and foreign table*

Designing Virtual Tables – Because virtual tables can be defined independently of the physical data objects, designers can design the virtual tables the way that is best for the applications.

Developers can define the virtual tables with their columns, data types, null specifications, and relationships. Informatica Data Services offers a diagramming interface to do this which makes designing virtual tables in Informatica Data Services an activity very much like designing tables with data modeling tools. Figure 9 shows the definition of a table called CUSTOMERS.

In fact, Informatica Data Services offers a complete data modeling environment in which a complete data model consisting of tables and interrelationships can be entered; see Figure 10.



CustSchema.CustomerTable

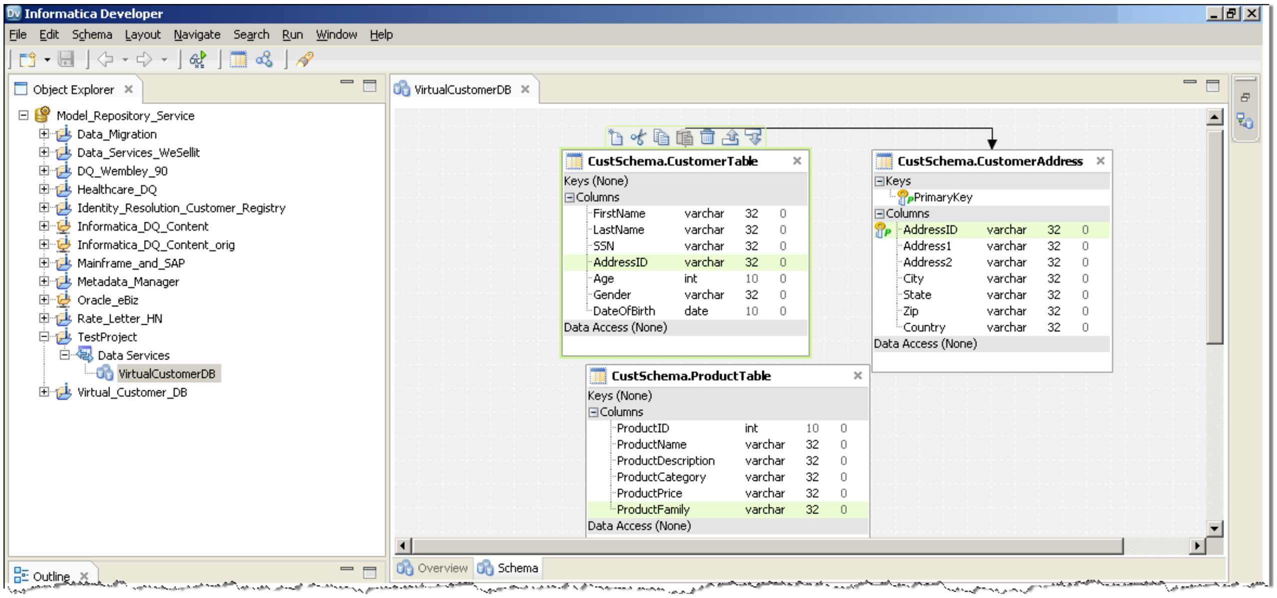
Keys (None)

Columns

FirstName	varchar	32	0
LastName	varchar	32	0
SSN	varchar	32	0
Age	int	10	0
Gender	varchar	32	0
DateOfBirth	date	10	0

Data Access (None)

Figure 9 *Defining a new virtual table*



CustSchema.CustomerTable

Keys (None)

Columns

FirstName	varchar	32	0
LastName	varchar	32	0
SSN	varchar	32	0
AddressID	varchar	32	0
Age	int	10	0
Gender	varchar	32	0
DateOfBirth	date	10	0

Data Access (None)

CustSchema.CustomerAddress

Keys

PrimaryKey

Columns

AddressID	varchar	32	0
Address1	varchar	32	0
Address2	varchar	32	0
City	varchar	32	0
State	varchar	32	0
Zip	varchar	32	0
Country	varchar	32	0

Data Access (None)

CustSchema.ProductTable

Keys (None)

Columns

ProductID	int	10	0
ProductName	varchar	32	0
ProductDescription	varchar	32	0
ProductCategory	varchar	32	0
ProductPrice	varchar	32	0
ProductFamily	varchar	32	0

Data Access (None)

Figure 10 *Defining a data model consisting of three tables and their relationships*

One advantage of separating a virtual table definition from a physical data object definition is that the latter can be used in the mappings for multiple virtual tables, or to put it differently, multiple virtual tables can share the same PDO specifications. Another advantage is that over time virtual tables can be "moved" to other PDO's. For example, when the real data is migrated to another database, only the mapping has to be redirected to another PDO. Or when a virtual table must be redirected from a foreign table in a testing environment to one in a production environment.

Instead of defining tables by hand, table definitions can also be imported from other tools that manage meta data specifications. Informatica Data Services uses Meta Integration's Model Bridge (MIMB) product to import those specifications. In addition, reverse engineering techniques can be applied to create these logical tables. The developer has to point to a database and Informatica Data Services will derive the table definitions and their relationships from the catalog of that database.

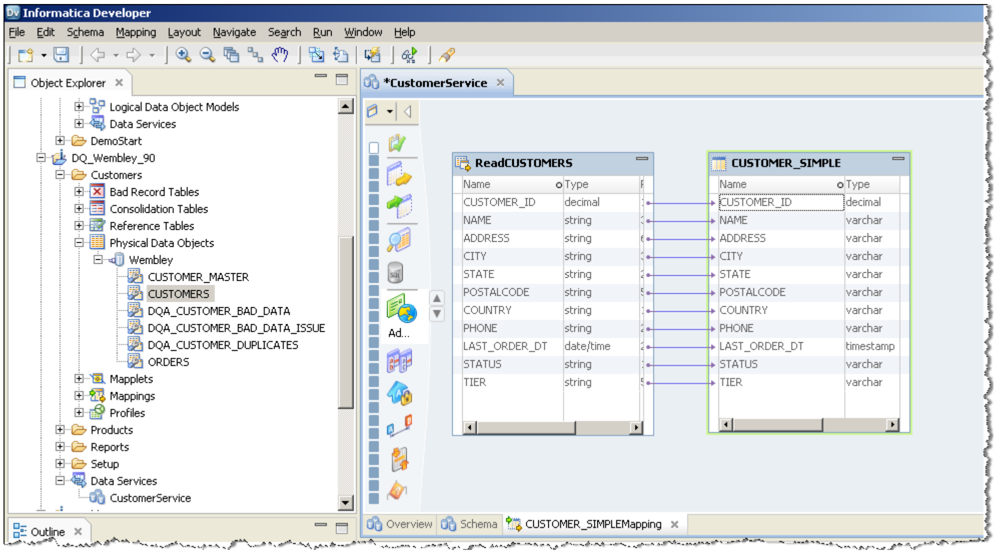
If a virtual table has been created and when it has been linked to a foreign table, accessing it is as simple as accessing a table in a SQL database. For example, if a tool uses SQL and an API, such as ODBC, JDBC, and OLE DB, Informatica Data Services will present the virtual table as a table. The reporting tool won't see the difference between an Informatica Data Services virtual table and a classic SQL-based database server table.

13 Defining Physical Data Objects

To make a virtual table queryable, it has to be linked to a physical data object. A physical data object is a set of specifications that describes the underlying foreign table. The PDO is like a wrapper. A foreign table can be, for example, a table in a SQL database server, an index sequential file, an XML document, or a spreadsheet. In this whitepaper we call them *foreign tables* to distinguish them from virtual tables. The specifications of PDO include column names, table names, data types, connection names, types of data source, and so on.

By creating a PDO, a foreign table becomes known to Informatica Data Services and therefore the foreign table becomes accessible for the applications using Informatica Data Services. The process to create a PDO for a foreign table is sometimes referred to as *importing a foreign table* and is a relatively easy exercise. Informatica Data Services can create a connection with the required database server and can retrieve meta data on tables from the catalog, such as column names, data types, null specifications, and population data. All this meta data is stored in the Informatica Data Services catalog. Once the foreign table has been imported and a PDO has been created, mappings can be defined on it to make it available for virtual tables; see the next section.

For testing or prototyping purposes, it's possible to let Informatica Data Services create automatically a virtual table plus a mapping on an imported PDO. The virtual contents of the virtual table will be identical to the real contents of the PDO. Each record in the foreign table will become one record in the virtual table, and all the columns of the foreign table are included 'as is' in the virtual table. In other words, a 1:1 mapping is created between the PDO and the virtual table. Figure 11 shows such a straightforward 1:1 mapping.



ReadCUSTOMERS

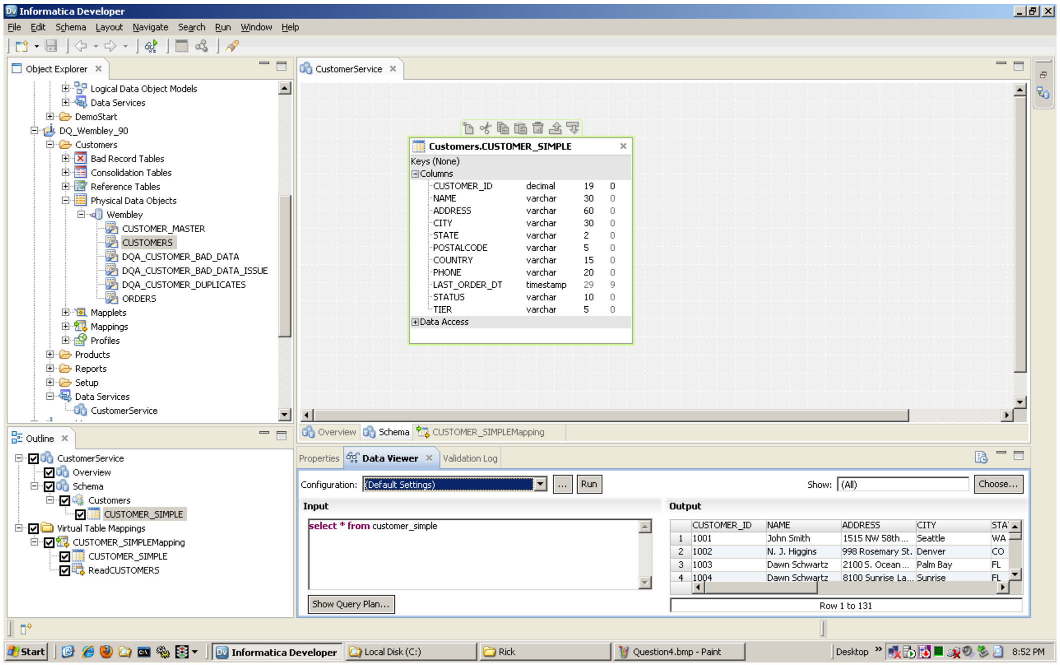
Name	Type
CUSTOMER_ID	decimal
NAME	string
ADDRESS	string
CITY	string
STATE	string
POSTALCODE	string
COUNTRY	string
PHONE	string
LAST_ORDER_DT	date/time
STATUS	string
TIER	string

CUSTOMER_SIMPLE

Name	Type
CUSTOMER_ID	decimal
NAME	varchar
ADDRESS	varchar
CITY	varchar
STATE	varchar
POSTALCODE	varchar
COUNTRY	varchar
PHONE	varchar
LAST_ORDER_DT	timestamp
STATUS	varchar
TIER	varchar

Figure 11 *An example of a 1:1 mapping linking a virtual table to a physical data object*

When a mapping has been created for a virtual table, it's accessible by any reporting tool. In fact, Informatica Data Services itself can already show its virtual contents; see Figure 12. This automatically created virtual table can be used, for example, to study the contents of the PDO. In the middle and at the bottom of the figure, a query on the virtual table CUSTOMERS is visible, and to the right the result of that query is displayed.



Customers.CUSTOMER_SIMPLE

Keys (None)

Columns

CUSTOMER_ID	decimal	19	0
NAME	varchar	30	0
ADDRESS	varchar	60	0
CITY	varchar	30	0
STATE	varchar	2	0
POSTALCODE	varchar	5	0
COUNTRY	varchar	15	0
PHONE	varchar	20	0
LAST_ORDER_DT	timestamp	29	9
STATUS	varchar	10	0
TIER	varchar	5	0

Data Access

Configuration: (Default Settings)

Show: (All)

Input

```
select * from customer_simple
```

Output

	CUSTOMER_ID	NAME	ADDRESS	CITY	STA
1	1001	John Smith	1515 NW 58th...	Seattle	WA
2	1002	N. J. Higgins	998 Rosemary St.	Denver	CO
3	1003	Dawn Schwartz	2100 S. Ocean...	Palm Bay	FL
4	1004	Dawn Schwartz	8100 Sunrise La...	Sunrise	FL

Row 1 to 131

Figure 12 *At the right-hand bottom side of the screenshot the virtual contents of a logical table is visible*

As indicated, physical data objects are not limited to SQL tables. In Section 20 other types of data sources are described.

14 Defining Data Object Mappings

A data object mapping defines how the data from a set of PDOs should be transformed to match the structure of a virtual table. Mappings can be a lot more sophisticated than the very simple 1:1 data object mapping shown in Figure 11. Data from multiple PDOs can be joined together, data can be aggregated, or particular column values can be transformed. All those and other transformations can lead to advanced mappings.

Each mapping consists of a number of transformation operations. Because the mapping in Figure 11 is simple, the mapping consists of just one operation. But a mapping can also consist of many operations. Examples of operations are filters, projection, groupings, joins, and de-duplications. For example, Figure 13 shows a mapping consisting of multiple operations (each symbol representing a transformation). The first symbol () indicates the PDO that is queried, in this case the CUSTOMER table. The second symbol () indicates a filter which means that a number of rows are selected. The third one () indicates that the values of a column are transformed. And finally, the fourth symbol () represents a grouping of rows which leads to aggregation of data.

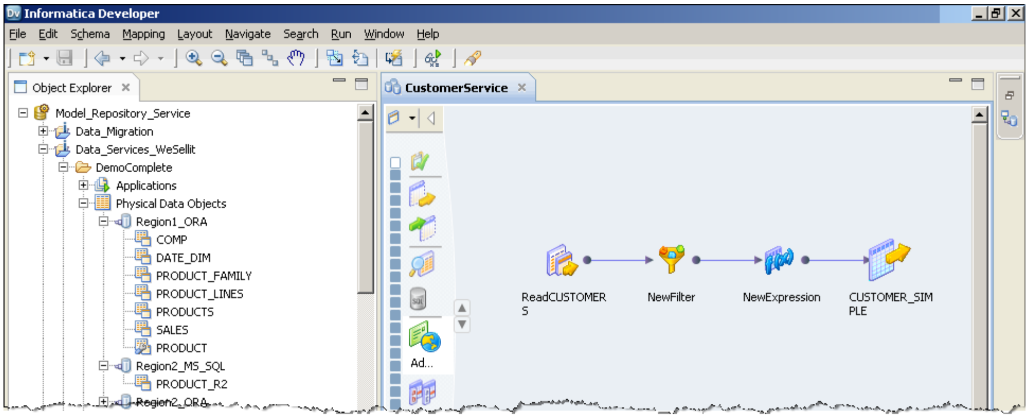


Figure 13 *An example of a more complex mapping linking a virtual table to a physical data object*

Note that this graphical language used for developing mappings is comparable to the one used in Informatica's ETL tool PowerCenter. So, developers familiar with this tool will find it easy to understand this aspect of Informatica Data Services.

Supported Transformation Operations – Here are some of the supported transformation operations:

- Filters can be specified to select a subset of all the rows from the foreign table.
- Columns in the foreign table can be removed from the logical table.
- Values can be transformed by applying a long list of string manipulation functions.
- Columns in the foreign table can be concatenated.
- Names of the columns in the foreign table and the name itself can be changed.

- New virtual and derivable columns can be added.
- Group-by operations can be specified to aggregate data.
- Statistical functions can be applied.
- Rows can be de-duplicated.
- Data can be cleansed.
- Rows can be sorted.
- Rank numbers can be assigned.

To show how advanced these mapping can be, Figure 14 contains an example of a more complex mapping in which many transformation operations are used.

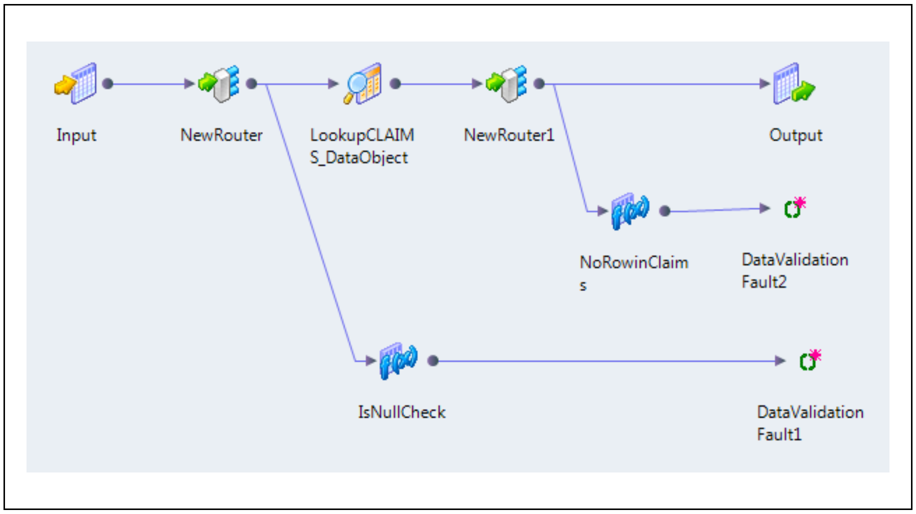


Figure 14 *An example of a complex mapping consisting of many transformation operations*

Home-made Transformations – If developers need transformation not supplied by Informatica Data Services, they can develop them themselves. They can use Java (including JDBC and SQL) for writing their own transformations, and they can write a transformation that consists of a SQL query that retrieves data from some data sources.

Reusable Transformation Logic – Developers can develop small transformations called *rules* or *maplets* that can be used and shared in multiple mappings. For example, a maplet can be written to remove all the non-digits from telephone numbers. Next, it can be used in all the mappings of tables that contain columns with telephone numbers. The advantage of this approach is that mapping logic can be re-used, which improves productivity and maintenance.

Scheduling the Mappings to run in Batch – Mappings are normally executed by Informatica Data Services in an on-demand way. Meaning, if the virtual is queried by an application, the mapping is processed; data is retrieved from the foreign tables, all the transformation operations are applied, and the result is passed to the application as a neat table consisting of rows and columns.

Besides processing mappings on-demand, their processing can also be scheduled. At certain time intervals the mapping is executed and the result of the mapping is not passed to a virtual table, but stored in a real table. In fact, the mapping is executed ETL-style. To get to this result, the virtual table should be defined on the table that holds the result of the mapping; see Figure 15. The advantage is that the same mapping code can be used to run the mapping in an on-demand and in a scheduled way.

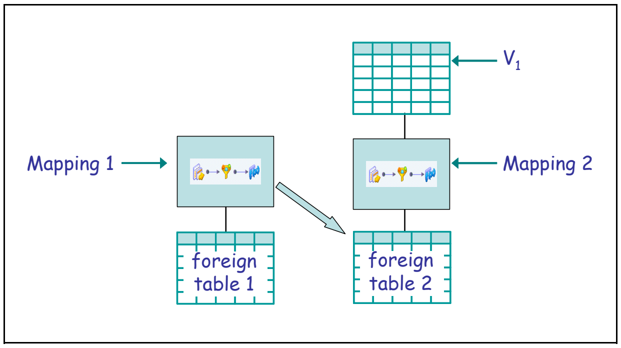


Figure 15 *Processing the mapping in batch*

Merging Data from Multiple Databases – Many reports need to combine data stored in multiple tables and even tables managed by different database systems. Informatica Data Services solves this by creating a mapping that integrates two or more foreign tables together. Those tables can be part of the same database, separate databases and even databases managed by different database servers. For example, Figure 16 shows the integration of tables stored in respectively an Oracle, DB2, and Netezza database. The mapping used can be as straightforward or as complex as the developer wants. He has the full pallet of transformation operations to his disposal to create the right mapping.

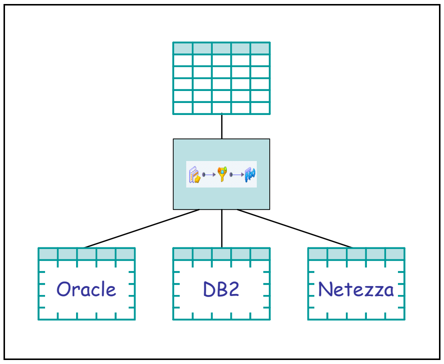


Figure 16 *Integrating data from different databases using a mapping*

15 Processing the Mappings by Translating them into SQL

Informatica Data Services has a unique way of processing the mappings. Whereas most ETL tools, including Informatica PowerCenter, natively process the transformation operations that make up the mappings, Informatica Data Services translates the mappings into SQL queries. Those SQL queries are processed by a new *SQL engine* that is part of Informatica Data Services and was developed by Informatica itself.

The advantage of this mapping language is that it supports transformation operations that standard SQL doesn't. For example, complex cleansing operations such as de-duplication, are not supported by most SQL products. The same applies for some other transformations. The developers can now exploit the full power of Informatica Data Services' mapping language.

One important task of the SQL engine is to come up with the best processing strategy for the SQL statement. In other words, the role of this engine is comparable to that of a query optimizer in a classic SQL database server.

One of the most important aspects is *push-down*. If a mapping accesses simple files, such as index sequential files or spreadsheets, Informatica Data Services has to do all the processing. In that case, the SQL engine inside Informatica Data Services will retrieve all the data and will run the SQL statement. However, if the tables accessed are managed by a database server, Informatica Data Services' SQL engine will try to push as much of the SQL processing to those underlying database servers. The SQL engine will be left with merging results coming from multiple database servers and with doing some extra transformations (the ones not supported by the database servers). This push-down concept is very important to improve performance and to minimize network traffic between Informatica Data Services and the database servers.

Note that SQL is hidden for the developers. They work with the mapping language. The translation to SQL is all done under the hood.

Besides the ability to push-down processing, Informatica Data Services offers a critical optimization form in a virtualized system: *early selection optimization*. This is the ability to push a filter (where clause in SQL) all the way into the source so the amount of data passed back to Informatica Data Services is much smaller than the entire data set. This will lead to fast responses even when the data source contains a large volume of data.

16 Sharing Specifications by Stacking Virtual Tables

Like views in a SQL database can be stacked, virtual tables in Informatica Data Services can be stacked as well. In other words, a virtual table can be defined on top of other virtual tables. A virtual table defined this way, is sometimes referred to as a *nested virtual table*. A nested virtual table is also defined using the mapping language. As an example, Figure 17 shows two nested virtual tables. Informatica Data Services allows virtual tables to be nested indefinitely.

Being able to stack virtual tables allows for shared specifications. Let's illustrate this with the situation shown in Figure 18. Here two nested virtual tables are defined on another virtual table. Let's assume that the two nested virtual tables are used by different applications and have different definitions. The advantage of this layered approach is that the specifications to be shared by both applications can be defined on virtual table V_1, and the specifications unique to the applications can be implemented in virtual tables V_2 and V_3. This means that the applications share common specifications. If we change those common specifications, that change will automatically apply to V_2 as to V_3.

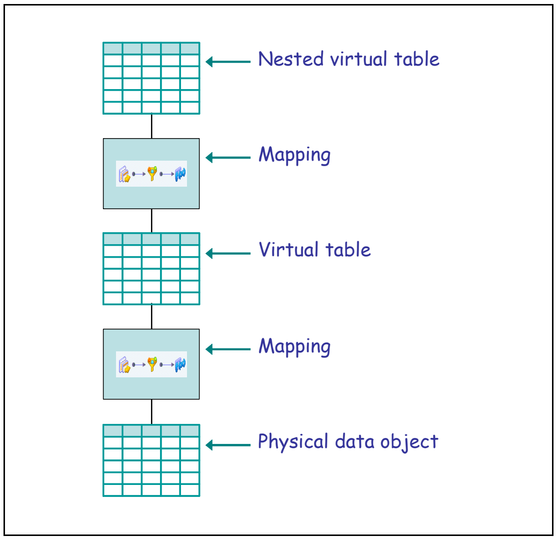


Figure 17 *At the top of the diagram are two nested virtual tables*

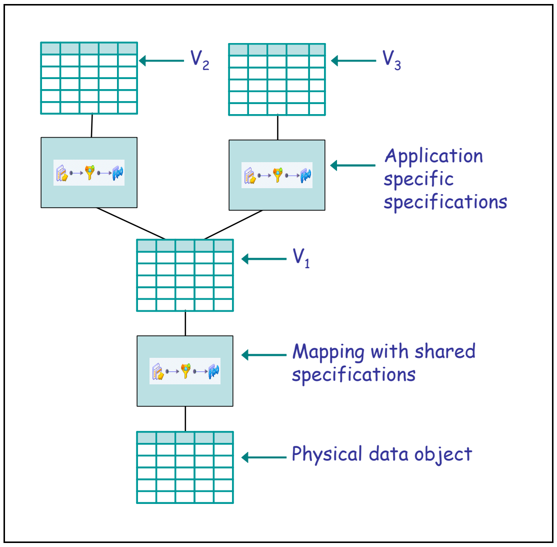

V_2
V_3
Application specific specifications
V_1
Mapping with shared specifications
Physical data object

Figure 18 *Shared specifications in virtual table V_1*

Example: Imagine an organization that groups the countries in which it operates in four regions. If they want to define that the Northern Region includes a specific set of countries, they can, for example, define that in the query that forms the body of a virtual table. This can be done by adding a filter to the query that guarantees that only specific countries appear. Every tool and every application accessing that virtual table will see that specific set of countries. These specifications become shareable and don't have to be repeated anymore inside the reporting and analytical tools themselves. This frees those tools to focus on their strengths, such as analytics, visualization, reporting, and drill-downs. So, even if these tools support light federation capabilities themselves, with this shared specifications approach, the limited functions won't be needed.

Shareable specifications will improve the consistency of reporting results, they will improve the speed of development and will ease maintenance: specifications only have to be updated once. For example, in the previous example, if the organization wants to move a country from one region to another, only once change is required and it will apply to all the reports and applications. In addition, it makes the use of a wide range of reporting tools possible without running the risk of inconsistent results.

17 Integrated and On-Demand Data Profiling

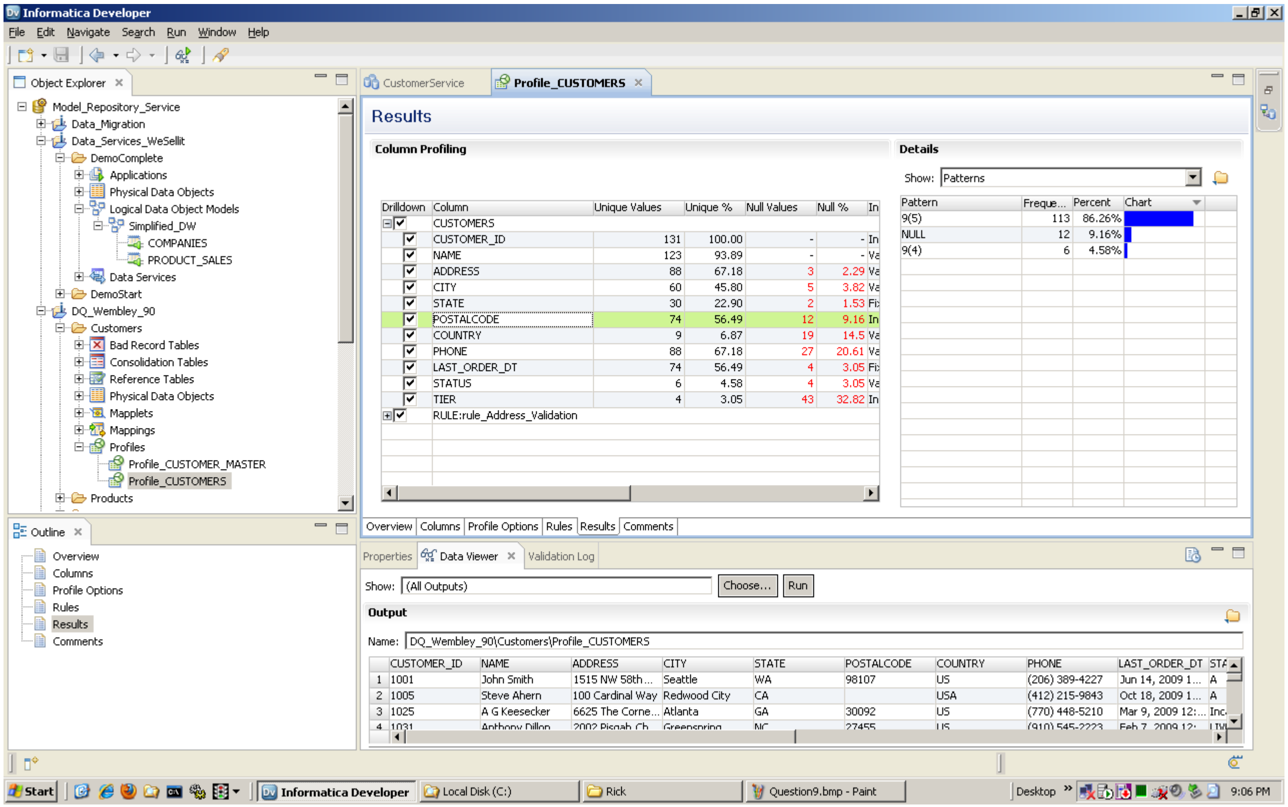
Data profiling is an exercise where data is analyzed to discover possible *defective data*. Defective data is data that is incorrect, missing, or inconsistent. If defective data has been identified, the organization has to make a decision on whether the defective data should be cleansed? This is usually a matter of comparing the costs of cleansing versus the benefits of having reports showing cleansed data.

Data Profiling Techniques – The data profiling module of Informatica Data Services supports a wide range of data profiling techniques that can be classified in two groups: *column profiling* and *join analysis techniques.* Column profiling techniques involve discovering patterns and extracting statistical data, such as the number of null values, the maximum and minimum value, the number of numeric and alphanumeric values, and so on. With join analysis the populations of two columns (possibly from different tables and data stores) are compared: do their populations overlap?

Let's begin with column profiling. Imagine that the developer has defined a virtual table on a foreign table and he expects only the values A, B, and C in a particular column. By using column profiling the developer can quickly and easily check whether those are really the only values in the underlying foreign table. This exercise could show that some undocumented values exist in that column which means the developer has to come up with a solution on how to handle those unexpected values. Should they be skipped or transformed?

Another example is presented in Figure 19 which contains the result of a column profiling exercise where statistical data is extracted from a column called POSTALCODE. You can clearly see that two different patterns exist in this column: one consisting of five digits and one consisting of four digits. Plus, there are a few null values in that column. It's up to the developer to determine whether all these values are correct or, for example, whether the four digit-values are incorrect. This form of column profiling is called *pattern analysis*.

Another example of column profiling would be if two foreign tables must be joined on two columns containing telephone numbers. By using column profiling the developer can check whether the values in those two columns really match. Maybe in one of the columns brackets are used in the telephone number, for example like this `(123)456789`, and the other uses the format `123-456789`. With profiling we can determine whether the columns are really joinable, or whether the values must be transformed first.



Results

Column Profiling

Drilldown	Column	Unique Values	Unique %	Null Values	Null %	In
⊟ ☑	CUSTOMERS					
☑	CUSTOMER_ID	131	100.00	-	-	In
☑	NAME	123	93.89	-	-	Va
☑	ADDRESS	88	67.18	3	2.29	Va
☑	CITY	60	45.80	5	3.82	Va
☑	STATE	30	22.90	2	1.53	Fi:
☑	POSTALCODE	74	56.49	12	9.16	In
☑	COUNTRY	9	6.87	19	14.5	Va
☑	PHONE	88	67.18	27	20.61	Va
☑	LAST_ORDER_DT	74	56.49	4	3.05	Fi:
☑	STATUS	6	4.58	4	3.05	Va
☑	TIER	4	3.05	43	32.82	In
⊞ ☑	RULE:rule_Address_Validation					

Details

Show: Patterns

Pattern	Freque...	Percent	Chart
9(5)	113	86.26%	
NULL	12	9.16%	
9(4)	6	4.58%	

Show: (All Outputs)

Output

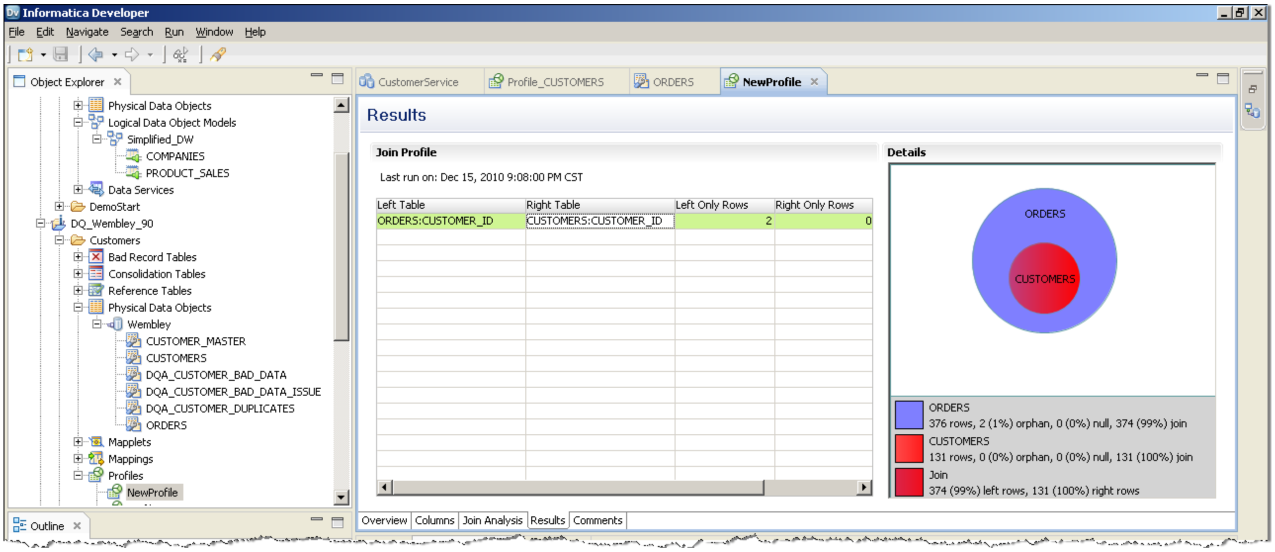
Name: DQ_Wembley_90\Customers\Profile_CUSTOMERS

	CUSTOMER_ID	NAME	ADDRESS	CITY	STATE	POSTALCODE	COUNTRY	PHONE	LAST_ORDER_DT	STA
1	1001	John Smith	1515 NW 58th...	Seattle	WA	98107	US	(206) 389-4227	Jun 14, 2009 1...	A
2	1005	Steve Ahern	100 Cardinal Way	Redwood City	CA		USA	(412) 215-9843	Oct 18, 2009 1...	A
3	1025	A G Keesecker	6625 The Corne...	Atlanta	GA	30092	US	(770) 448-5210	Mar 9, 2009 12:...	Inc.
4	1031	Anthony Dillon	2002 Pisgah Ch...	Greenspring	NC	27455	US	(910) 545-2223	Feb 7, 2009 12:...	LIV

Figure 19 *The result of a column profiling exercise*

Another data profiling technique is *join analysis*. Imagine that a developer wants to join the foreign tables CUSTOMERS and ORDERS on the columns called CUSTOMER_ID. If the join analysis indicates that there are values in CUSTOMER_ID of the ORDERS table that don't appear in CUSTOMERS, it means that there are orders with no matching customer, which must be incorrect. Now the developer knows this and can act accordingly. The users themselves must fix it by connecting those dangling orders to the right customers, by deleting them, or by inserting the missing customers. If they don't or can't fix this inconsistency the developer has to come up with a solution. The transformations should probably skip those dangling orders. Figure 20 presents the result of such an join analysis. The Venn diagram at the top right-hand side of the figure shows that there is no problem, because the values in the ORDERS table all appear in the CUSTOMERS table.

On-Demand Data Profiling – Note that all the data profiling functionally works directly on the virtual tables. If the profiling functionality accesses those virtual tables, only then is the data to be profiled retrieved from the data stores. There is no need to extract data from a data store first, file it somewhere else, and then activate the profiling. In other words, Informatica Data Services supports *on-demand data profiling*.



Join Profile

Last run on: Dec 15, 2010 9:08:00 PM CST

Left Table	Right Table	Left Only Rows	Right Only Rows
ORDERS:CUSTOMER_ID	CUSTOMERS:CUSTOMER_ID	2	0

🟦 ORDERS
376 rows, 2 (1%) orphan, 0 (0%) null, 374 (99%) join

🟥 CUSTOMERS
131 rows, 0 (0%) orphan, 0 (0%) null, 131 (100%) join

🟥 Join
374 (99%) left rows, 131 (100%) right rows

Figure 20 *The result of a join analysis is presented as a Venn diagram*

***Integrated Data Profiling* –** Most current data profiling tools work offline. They are hooked-up to a database and are guided by the analyst to profile the data. They are not integrated with an ETL or a federation tool. In most cases the effect is that developers must go back and forth between the profiling tool and their integration tool.

In Informatica Data Services data profiling is an integral part of the development environment. When foreign tables have been imported within Informatica Data Services, developers can activate the data profiling module to verify whether the data in the foreign table(s) is as expected and correct. In addition, when virtual tables and mappings have been defined, developers can also study the virtual contents of the virtual tables to see whether the mappings have the desired result. Literally, the developer can profile the virtual data and source data, change the mapping, and he can see the impact on the outcome of the new mapping instantly. In other words, Informatica Data Services supports *integrated data profiling*. There is no loss of time due to going back and forth between tools.

18 Collaborative and On-Demand Data Cleansing

After defective data has been detected and the organization has decided it should be cleansed, the decision must be made where and how the defective data should be cleansed. Should the data source be cleansed (outside the scope of the data warehouse environment) or should the data source stay unchanged and should the transformation process change the data from defective to correct data (inside the scope of the data warehouse environment)?

***On-Demand Cleansing* –** If the data should be cleansed inside the data warehouse environment, an option is that developers implement cleansing logic inside the mappings to cleanse the data. For example, incorrect codes can be replaced by correct ones, orders not linked to customers can be skipped, and tables should be joined using a fuzzy match algorithm. Beside using classic string manipulations and other operations, Informatica Data Services does support a set of special operations for cleansing.

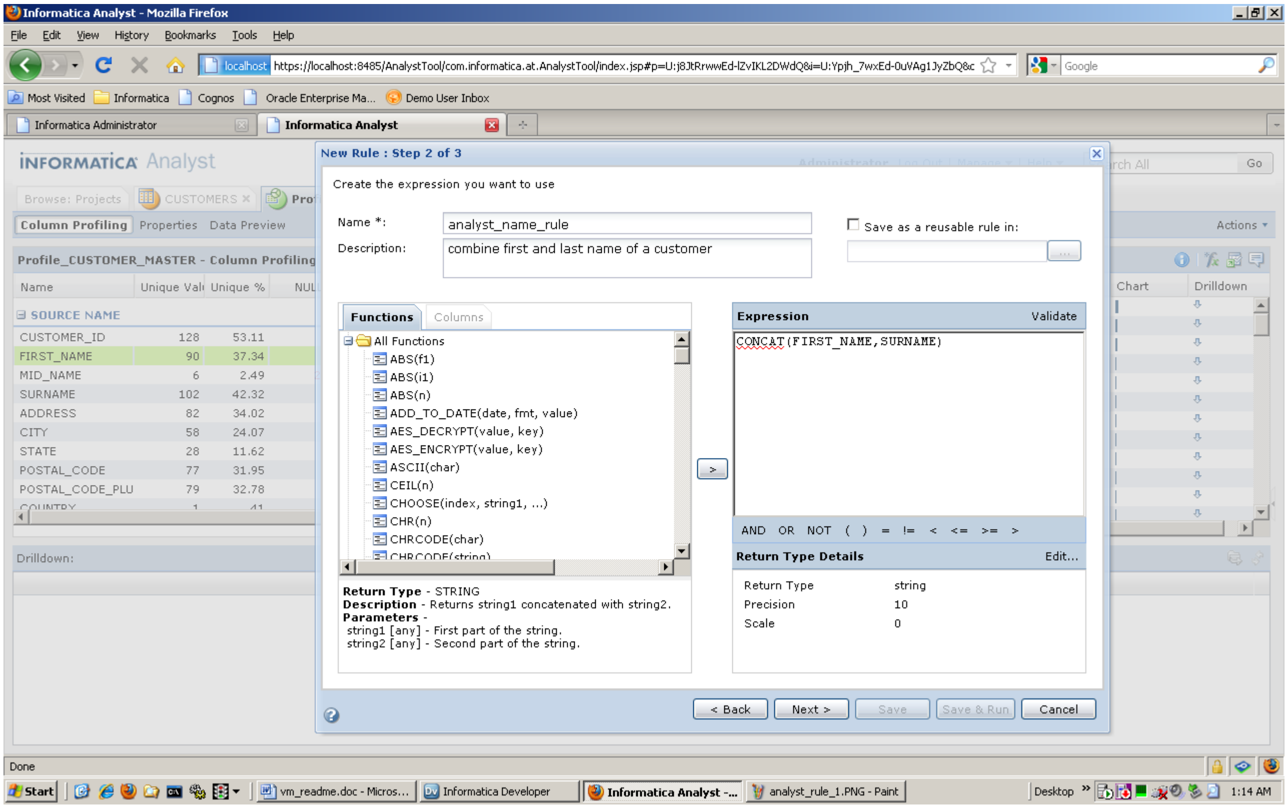
Because the data is cleansed in “flight”, meaning the result is not stored, this form of cleansing is called *on-demand data cleansing*. The data is cleansed the moment it is retrieved by the reports and applications.

The advantage of this approach is that no need exists to create and manage additional data stores. When the cleansing operations take too long, it can always be decided to create a cache for the virtual table that presents the cleansed data; see Section 23 for a description of caching.

Collaborative Data Cleansing – In reality, developers can’t always decide by themselves how defective data should be transformed. Business knowledge is required. For example, if the column GENDER is only supposed to contain the codes `M` and `F`, the developer will know what to do with the incorrect value `m`: it should be capitalized. But what if the value `X` appears in this column of a production system? Was this letter used instead of `M` or `F`, or is it something special, maybe a new value? This is a difficult question to answer for a developer who hasn’t been involved in entering the data. Or what should be done if two customer records are very much identical? Should they be merged into one record? It’s clear that input of business analysts is inevitable. However, if every time a piece of defective data arises requires that developers have to meet with the analysts to determine what to do, the entire data profiling exercise will become a very time-consuming process.

To optimize this process, Informatica Data Services supports collaboration: developers and business analysts can work together on the mappings to get the data right. It allows analysts to become involved in the profiling process. Developers can ask analysts whether the right transformations are being applied. Using a different Informatica Data Services module, one that is web-based and specifically designed for non-technical users, they can study the contents of a virtual table and check whether it is according to what they expect. Figure 21 shows a screenshot of what the user interface looks like. If the transformations are not correct, analysts can give the developers advice, or they can develop the correct transformations themselves. The language used here strongly resembles the language used in Excel (which is probably a language most analysts are familiar with). It’s a language consisting of mathematical and string manipulation functions. Transformations developed by analysts can be used by developers in the real mappings. Conclusion, because of this feature, data cleansing becomes a collaborative process where developers and analysts work together to get the data in the form the business requires.

By making it a collaborative exercise, there will be less going back and forth, which should reduce the time it takes to get all the mappings defined. In addition, this collaboration increases the chance that developers deliver a result that the analysts agree with.


Informatica Analyst - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Most Visited Informatica Cognos Oracle Enterprise Ma... Demo User Inbox
Informatica Administrator
Informatica Analyst
INFORMATICA Analyst
Browse: Projects CUSTOMERS Pro
Column Profiling Properties Data Preview
Drilldown:
Actions
Chart Drilldown
Functions Columns
AND OR NOT ( ) = != < <= >= >
< Back Next > Save Save & Run Cancel
Done
Start
vm_readme.doc - Micros...
Informatica Developer
Informatica Analyst -...
analyst_rule_1.PNG - Paint
Desktop
1:14 AM

https://localhost:8485/AnalystTool/com.informatica.at.AnalystTool/index.jsp#p=U:j8JtRrwwEd-lZvIKL2DWdQ8i=U:Ypjh_7wxEd-0uVAg1JyZbQ8c

Profile_CUSTOMER_MASTER - Column Profiling

Name	Unique Val	Unique %
SOURCE NAME		
CUSTOMER_ID	128	53.11
FIRST_NAME	90	37.34
MID_NAME	6	2.49
SURNAME	102	42.32
ADDRESS	82	34.02
CITY	58	24.07
STATE	28	11.62
POSTAL_CODE	77	31.95
POSTAL_CODE_PLU	79	32.78

New Rule : Step 2 of 3

Create the expression you want to use

Name *: analyst_name_rule

Description: combine first and last name of a customer

□ Save as a reusable rule in:

All Functions
- ABS(f1)
- ABS(i1)
- ABS(n)
- ADD_TO_DATE(date, fmt, value)
- AES_DECRYPT(value, key)
- AES_ENCRYPT(value, key)
- ASCII(char)
- CEIL(n)
- CHOOSE(index, string1, ...)
- CHR(n)
- CHRCODE(char)

Return Type - STRING
Description - Returns string1 concatenated with string2.
Parameters -
string1 [any] - First part of the string.
string2 [any] - Second part of the string.

Expression Validate

```
CONCAT(FIRST_NAME,SURNAME)
```

Return Type Details Edit...

Return Type	string
Precision	10
Scale	0

Figure 21 *The interface for an analyst for studying the definitions and contents of virtual tables and where his own transformations can be entered*

19 Developing Virtual Data Marts

The structures of foreign tables have typically been designed to support the original production applications. But usually that means they don't have the right structure for reporting and analytics. For example, some reporting tools prefer the tables to have a *star schema* arrangement, others favor a *snowflake schema* arrangement, and a third group likes to see the data fully normalized[7]. Sometimes performance is the reason why other structures are selected than the original ones of the foreign tables.

To solve this problem in classic architectures, a separate data store is created in which tables are designed in such a way that they meet the needs of the reports. Such a data store is normally referred to as a *data mart*. The disadvantages of this solution is that a data mart is quite an investment on itself, it must be designed, managed, and periodically refreshed.

In Informatica Data Services a more flexible and lightweight solution can be developed where the data marts consist of a number of virtual tables. This is sometimes called a *virtual data mart*.

[7] M. Golfarelli and S. Rizzi, *Data Warehouse Design – Modern Principles and Methodologies*, McGraw-Hill, 2009.

We will illustrate this with a simple example.

Imagine that some foreign tables, that belong to a number of old production systems, don't have the right column structures for our new reporting and analytics needs. Virtual tables with the right structure can be developed on top of those foreign tables. This can be done with one layer of mappings and nested virtual tables. Figure 22 presents such a solution. Layer 1 contains the foreign tables, layer 2 contains the PDOs, Layer 3 shows the virtual tables with their starschema arrangements, and Layer 4 presents the applications. For each virtual table a mapping exists, but for simplicity sake they have been left out.

In a way, these virtual tables on Layer 3 together form a data mart. Normally, a data mart contains a set of tables that have a structure and content that fits the reports accessing them. With this solution the tables are virtual. In other words, no data is stored. Together these virtual tables form a *virtual data mart*. As can be imagined, this solution is easier to develop, manage, and maintain than a physical data mart. In other words, it will be more cost-effective.

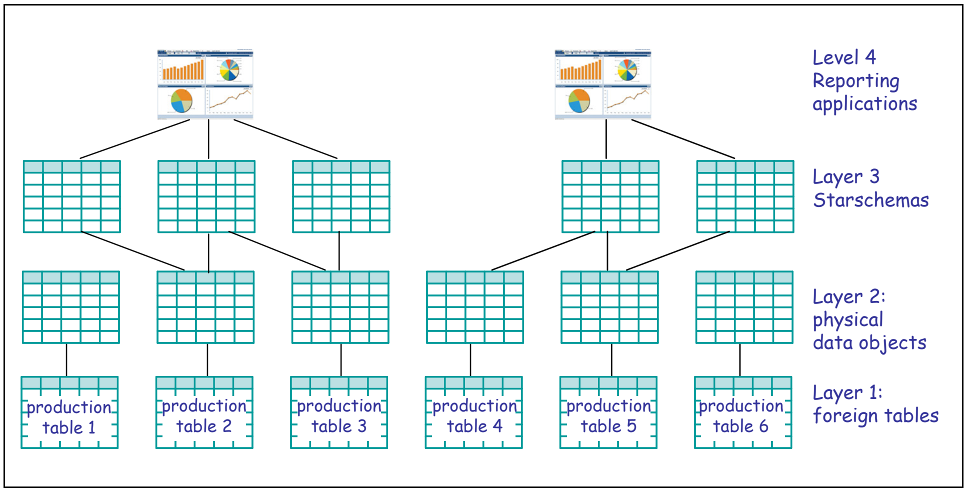


Figure 22 *A set of virtual tables forms a virtual data mart*

The above solution represents one way of creating a virtual data mart in Informatica Data Services. The virtual tables probably all use complex mappings to transform the structure and data of the foreign tables to those of the virtual tables. Because each virtual table has its own mapping, this solution might lead to a lot of repetition of logic in the mappings. By introducing more layers of virtual tables more mapping logic can be reused. Let's illustrate this as well.

We could start with defining a set of virtual tables that have a normalized structure; see Layer 3 in Figure 23. These mappings might contain all the necessary cleansing operations. Next, on top of those a set of virtual tables is defined containing transformations that apply to all the users; this would be Layer 4 in that same figure. Subsequently, for each report or tool a separate set of virtual tables is defined that presents the data in a suitable form; Layer 5. This could mean that one set of virtual tables has a star schema arrangement, the other a snowflake schema arrangement, and the third a classic normalized arrangement. This might all be dependent on the tools used by the users. In other words, Layer 5 contains the specifications that are specific to a user, report, or tool, whereas Layers 3 and 4 contain shareable specifications that apply to all the users.

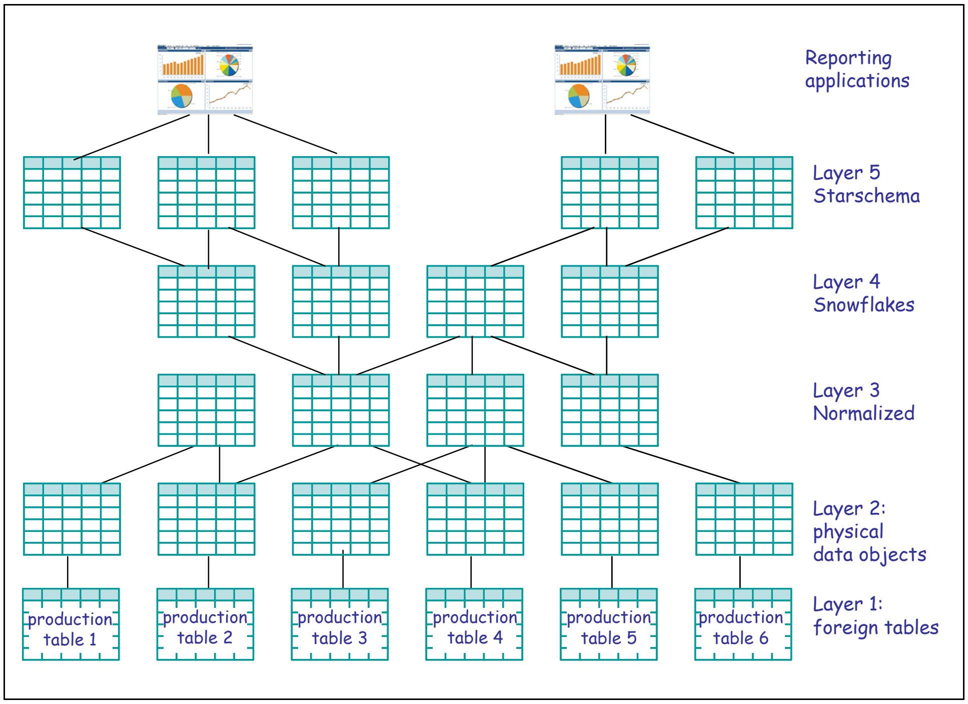


Figure 23 *Levels of virtual tables; for simplicity mappings are omitted in this diagram*

Another option could be that the views on Layer 3 are modeled according to the design principles of *Data Vault*[8], and that Layer 4 contains the more user specific specifications.

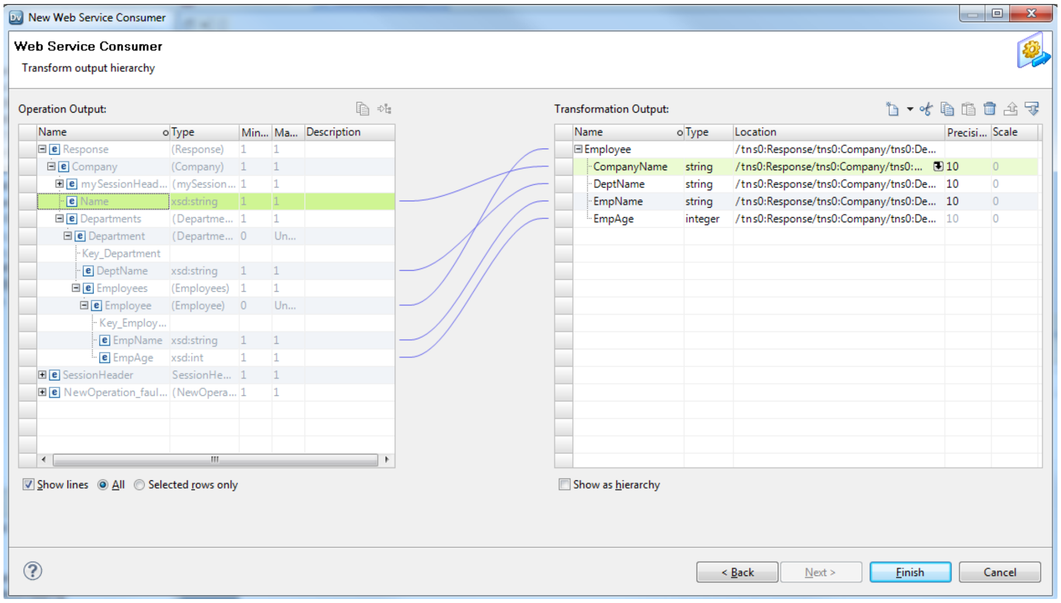
To summarize, developers are completely free in designing a structure that fits the needs of the user. This virtual solution is easy to change, and if the right design techniques are applied, many mapping specifications can be reused. Virtual data marts are an extremely flexible solution and incredibly cost-effective specially compared to its physical counterpart.

20 Transforming XML Documents and Spreadsheets to Tables

Not all data needed for analytics and reporting is stored in relational tables. It might be stored in all kinds of data stores and formats. This section describes how data stored in XML documents and spreadsheets can be imported into Informatica Data Services and becomes accessible as tables.

From XML to Tables – Before data in an XML document can be used, it has to be *flattened*, or in other words, the hierarchical structure of the XML document has to be turned into a flat relational table. In Informatica Data Services, the mechanism to do is to define a mapping for the document; see Figure 24.

[8] See http://en.wikipedia.org/wiki/Data_Vault_Modeling.



Web Service Consumer

Transform output hierarchy

Operation Output:

Name	Type	Min...	Ma...	Description
⊟ Response	(Response)	1	1	
⊟ Company	(Company)	1	1	
⊞ mySessionHead...	(mySession...	1	1	
Name	xsd:string	1	1	
⊟ Departments	(Departme...	1	1	
⊟ Department	(Departme...	0	Un...	
Key_Department				
DeptName	xsd:string	1	1	
⊟ Employees	(Employees)	1	1	
⊟ Employee	(Employee)	0	Un...	
Key_Employ...				
EmpName	xsd:string	1	1	
EmpAge	xsd:int	1	1	
⊞ SessionHeader	SessionHe...	1	1	
⊞ NewOperation_faul...	(NewOpera...	1	1	

☑ Show lines ● All ○ Selected rows only

Transformation Output:

Name	Type	Location	Precisi...	Scale
⊟ Employee		/tns0:Response/tns0:Company/tns0:De...		
CompanyName	string	/tns0:Response/tns0:Company/tns0:...	10	0
DeptName	string	/tns0:Response/tns0:Company/tns0:De...	10	0
EmpName	string	/tns0:Response/tns0:Company/tns0:De...	10	0
EmpAge	integer	/tns0:Response/tns0:Company/tns0:De...	10	0

□ Show as hierarchy

Figure 24 *Transforming a hierarchical XML structure to a flat table*

In this figure, the field `Operation Output` contains the structure of the XML document, and the field called `Transformation Output` contains the required table structure. A transformation is specified by linking elements from the XML document to columns in the data source (curly lines). The lines indicate how the hierarchical structure of the XML document should be mapped to a table with columns.

As with the foreign tables, to make data available for access, a virtual table must be defined for the mapping; see Figure 25. If certain rows of the document must be left out, or if certain transformations must be applied, they can be defined in the mapping.

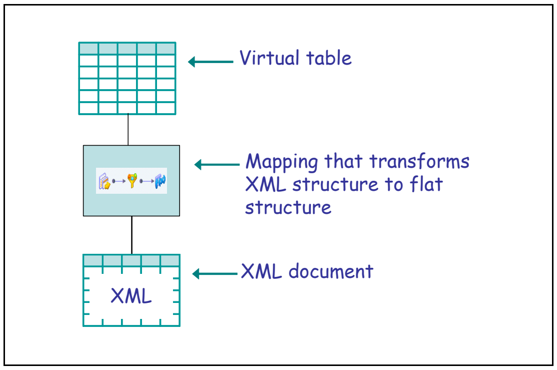


Figure 25 *Relational view on an XML document*

This virtual table on top of an XML document can be used like any other virtual table, so, for example, data stored in XML documents can be joined with data stored in relational tables; see Figure 26.

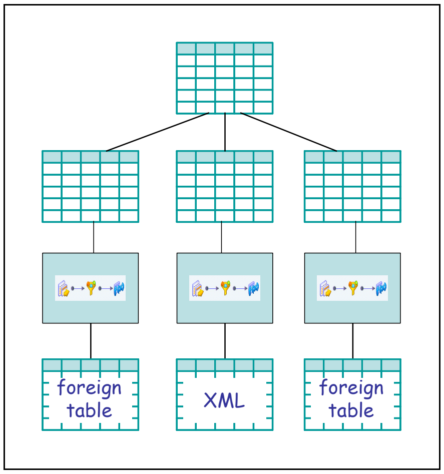


Figure 26 *Joining relational data with XML data*

Note: An XML structure doesn't always have to be flattened before it becomes usable within Informatica Data Services. It depends on what the consuming interface wants to expose. For example, in the case of a SQL query or virtual table consuming interface, the data must be flattened since SQL does not support hierarchical data structure queries. However, if the consuming interface is a web service, it is possible to go through the entire process without flattening the XML structure or hierarchy.

From Spreadsheets to Tables – Data that must be analyzed might also be stored in Microsoft Excel spreadsheets. In most cases, this is private data. Informatica Data Services makes it possible to combine the spreadsheet data with other data. To access spreadsheet data in Informatica Data Services or to join spreadsheet data with other data stores, the B2B DT transformation is used. This is an integrated part of Informatica Data Services for accessing all semi-structured and unstructured data sources, such as Excel, Word, PDF, and EDI.

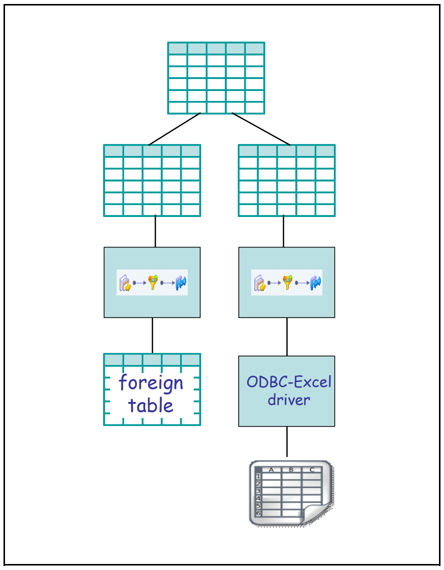


Figure 27 *Joining relational data with spreadsheet data*

21 Keeping Track of All Relationships

A large environment may end up with many definitions of foreign tables, mappings, virtual tables, and many interrelationships. It's important that developers and administrators can easily view all those relationships. This might be necessary, for example, to determine what the effect will be if the structure of a foreign table changes: which mappings and virtual tables must be changed accordingly?

Informatica Data Services stores all the definitions of foreign tables, mappings, virtual tables, and so on, in one central repository. This makes it easy for Informatica Data Services to show all the inter-dependencies between those objects. For example, Figure 28 shows such a dependency diagram. This is sometimes called a *lineage diagram*. Such a diagram allows us to do *impact analysis*. If the structure of one object changes, we can see on which other objects it might have an impact.

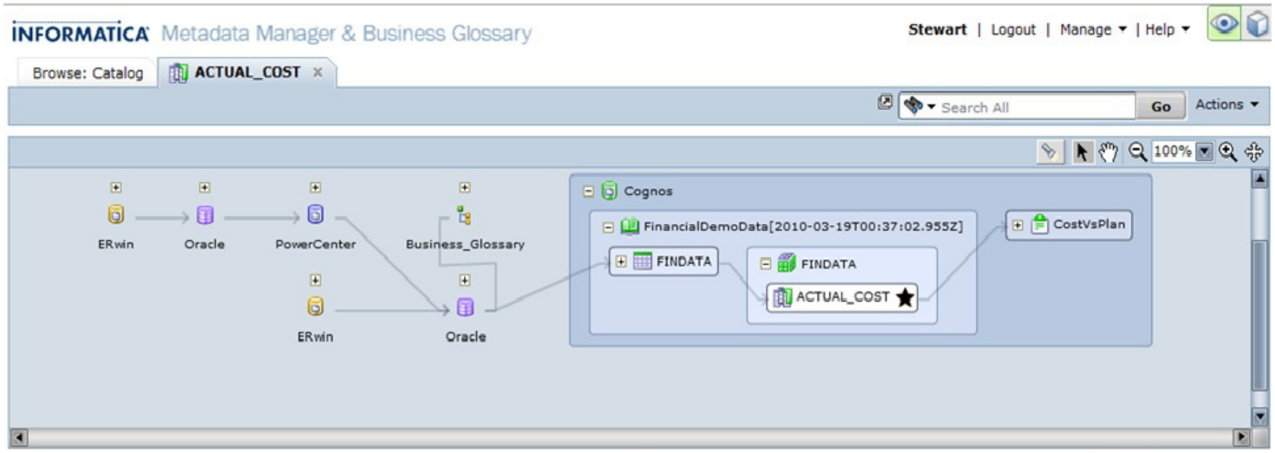


Figure 28 *A lineage diagram that shows the inter-dependencies between objects*

When objects such as physical data objects, mappings, and virtual tables are interrelated, what will happen if the underlying foreign table is changed? For example, what happens if a column is added to a foreign table, if one is removed, if the data type of a column is changed, or if a column is renamed? The solution is *synchronization*. With Informatica Data Services' synchronize function changes are automatically picked up and developers are allowed to modify the specifications accordingly.

22 Exposing Virtual Tables as Web Services

Most virtual tables in Informatica Data Services will be accessed by reporting tools using SQL. But Informatica Data Services also allows the tables to be accessed as *web services*. In this case the virtual tables are exposed as SOAP- or REST-based services. This can be done for every virtual table defined in Informatica Data Services, including the ones defined on XML documents, spreadsheets, and files. The advantage is that non-BI consumers, such as internet applications, service oriented architectures, and more classic data entry applications, can also access the data made available to the BI environment; see Figure 29. They will share the same specifications, and data will be consistent across the BI and other environments.

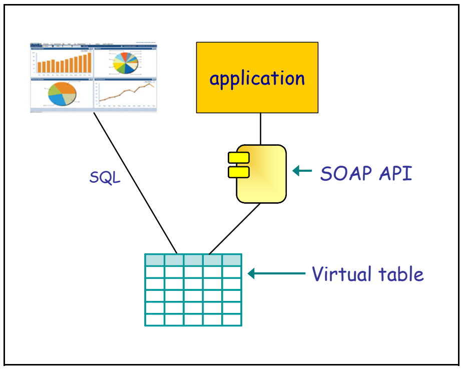
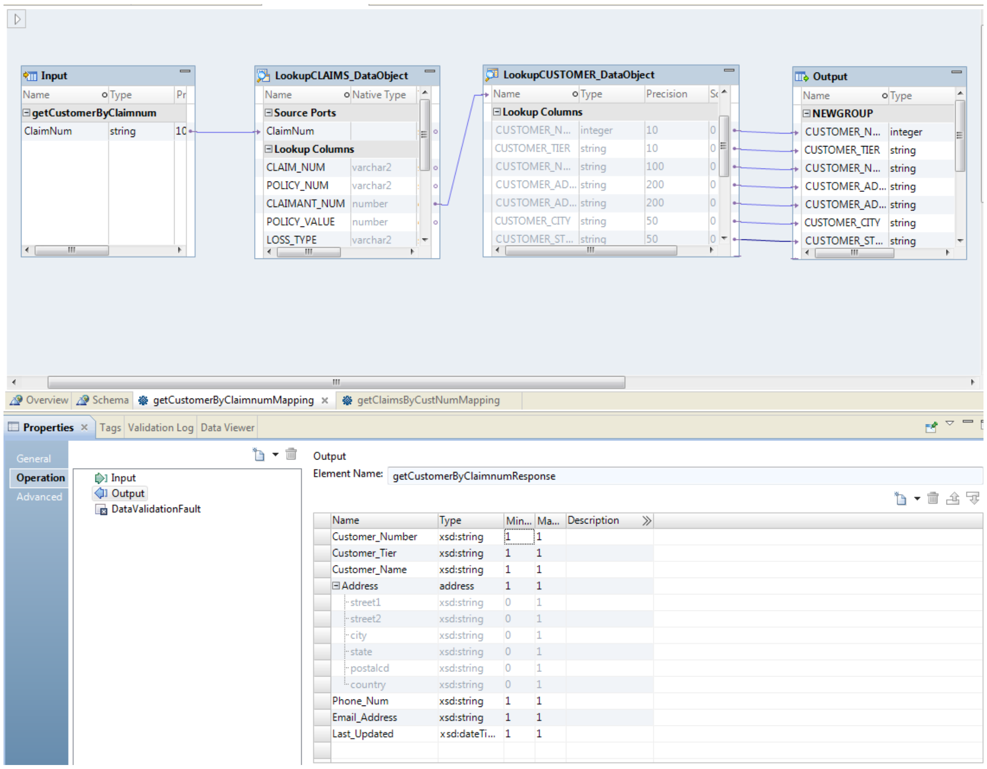


Figure 29 *Reports and applications share the same virtual table definitions*

To expose a virtual table as a web service we must define how the column structure of the virtual table must be transformed to an XML structure; see Figure 30. The top part of this figure contains the specification of the tables being accessed and the bottom part shows the interface of the service. In this example, the result has a hierarchical structure consisting of seven elements of which Address has six sub-elements.

Note that defining a transformation in this case of creating a web service interface is not mandatory. If we want to create a web service that has a flat XML structure, Informatica Data Services can automatically derive a flat XML structure from the virtual table's structure. However, if we want to bring some hierarchy to the structure and make it look like a real XML structure, we must define a transformation.

Next, an XML schema must be generated for this web service; see Figure 31. In addition, the WSDL document must be generated. When this step is completed, the web service is ready to be accessed. Any application that can access a web service can access a Informatica Data Services data service. Informatica Data Services itself will process the service requests.



Input

Name	Type	Pr
getCustomerByClaimnum		
ClaimNum	string	10

LookupCLAIMS_DataObject

Name	Native Type
Source Ports	
ClaimNum	
Lookup Columns	
CLAIM_NUM	varchar2
POLICY_NUM	varchar2
CLAIMANT_NUM	number
POLICY_VALUE	number
LOSS_TYPE	varchar2

LookupCUSTOMER_DataObject

Name	Type	Precision	Sc
Lookup Columns			
CUSTOMER_N...	integer	10	0
CUSTOMER_TIER	string	10	0
CUSTOMER_N...	string	100	0
CUSTOMER_AD...	string	200	0
CUSTOMER_AD...	string	200	0
CUSTOMER_CITY	string	50	0
CUSTOMER_ST...	string	50	0

Output

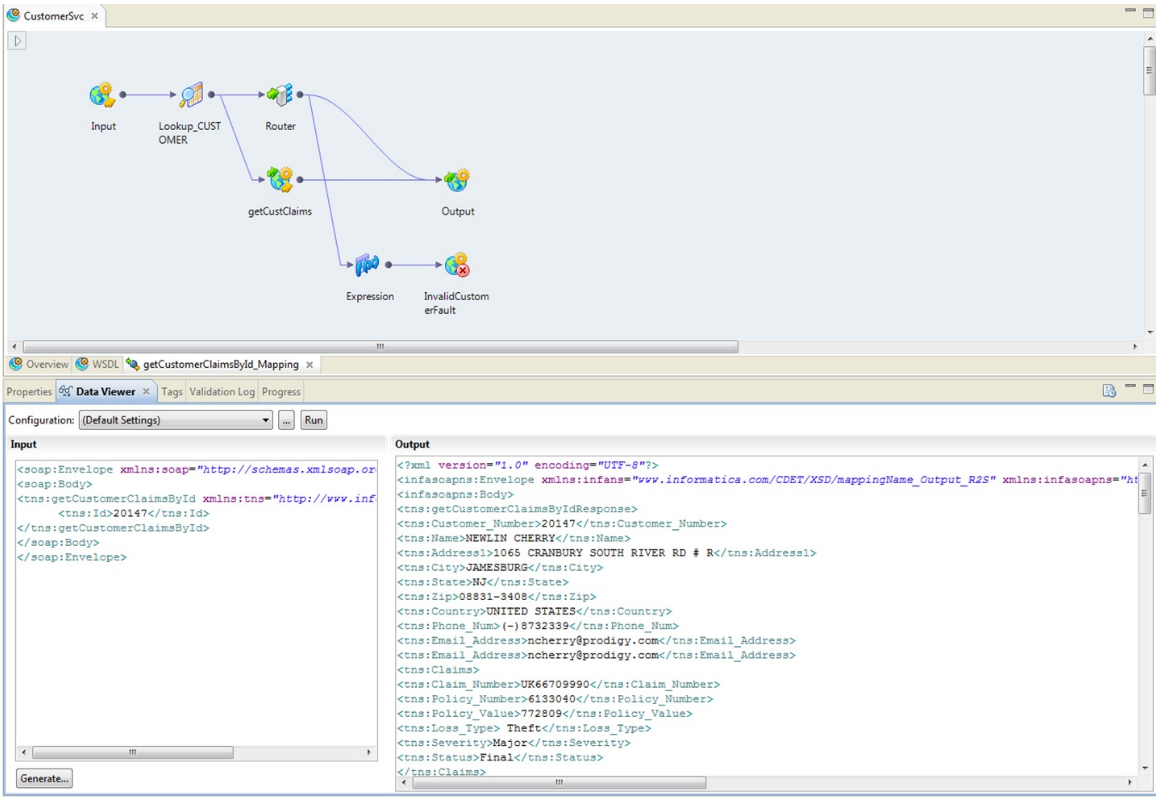
Name	Type
NEWGROUP	
CUSTOMER_N...	integer
CUSTOMER_TIER	string
CUSTOMER_N...	string
CUSTOMER_AD...	string
CUSTOMER_AD...	string
CUSTOMER_CITY	string
CUSTOMER_ST...	string

Output

Element Name: getCustomerByClaimnumResponse

Name	Type	Min...	Ma...	Description
Customer_Number	xsd:string	1	1	
Customer_Tier	xsd:string	1	1	
Customer_Name	xsd:string	1	1	
Address	address	1	1	
street1	xsd:string	0	1	
street2	xsd:string	0	1	
city	xsd:string	0	1	
state	xsd:string	0	1	
postalcd	xsd:string	0	1	
country	xsd:string	0	1	
Phone_Num	xsd:string	1	1	
Email_Address	xsd:string	1	1	
Last_Updated	xsd:dateTi...	1	1	

Figure 30 *Mapping a logical table to an XML structure*



Configuration: (Default Settings) ... Run

Input

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.or
<soap:Body>
<tns:getCustomerClaimsById xmlns:tns="http://www.inf
      <tns:Id>20147</tns:Id>
</tns:getCustomerClaimsById>
</soap:Body>
</soap:Envelope>
```

Output

```
<?xml version="1.0" encoding="UTF-8"?>
<infasoapns:Envelope xmlns:infans="www.informatica.com/CDET/XSD/mappingName_Output_R2S" xmlns:infasoapns="ht
<infasoapns:Body>
<tns:getCustomerClaimsByIdResponse>
<tns:Customer_Number>20147</tns:Customer_Number>
<tns:Name>NEWLIN CHERRY</tns:Name>
<tns:Address1>1065 CRANBURY SOUTH RIVER RD # R</tns:Address1>
<tns:City>JAMESBURG</tns:City>
<tns:State>NJ</tns:State>
<tns:Zip>08831-3408</tns:Zip>
<tns:Country>UNITED STATES</tns:Country>
<tns:Phone_Num>(-)8732339</tns:Phone_Num>
<tns:Email_Address>ncherry@prodigy.com</tns:Email_Address>
<tns:Email_Address>ncherry@prodigy.com</tns:Email_Address>
<tns:Claims>
<tns:Claim_Number>UK66709990</tns:Claim_Number>
<tns:Policy_Number>6133040</tns:Policy_Number>
<tns:Policy_Value>772809</tns:Policy_Value>
<tns:Loss_Type> Theft</tns:Loss_Type>
<tns:Severity>Major</tns:Severity>
<tns:Status>Final</tns:Status>
</tns:Claims>
```

Figure 31 *The generated XML schema of a web service*

23 Caching Virtual Tables

Regardless of how efficient a federation server such as Informatica Data Services is, it's an extra layer of software that sits between the reporting tools and the data stores, so it consumes cpu cycles and it increases the response time of queries executed by applications. Although we must say that the performance of a query is determined by the amount of time used by the federation server plus the time used by the underlying database server(s), the latter will consume most of the processing time, and the former only a small fraction. Still, it's important that a federation server optimizes and improves the performance of queries as much as possible. In this and the following sections a few of the techniques deployed by Informatica Data Services are described. This section explains *caching* of virtual tables.

Informatica Data Services offers an extensive and flexible caching mechanism. In Informatica Data Services caching means that the contents of a virtual table is retrieved from the underlying tables and stored in a file or table. The effect is that when the virtual table is queried the underlying foreign table, virtual table, file, or XML document is not accessed, but the data in the cache is. The main advantage of accessing a cache is that it can seriously improve the performance of a query on a virtual table.

For every virtual table a cache can be defined. Defining a cache involves nothing more than switching it on for a virtual table. Technically it means that the mappings and queries that make up the definition of the virtual table are executed and the final result is stored instead of passed on to an application. The next time this virtual table is queried, the data is retrieved from the cache; see Figure 32.

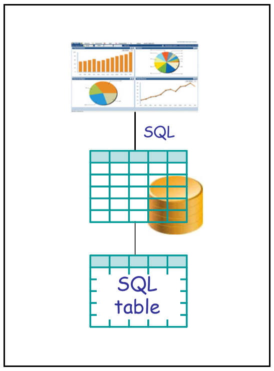
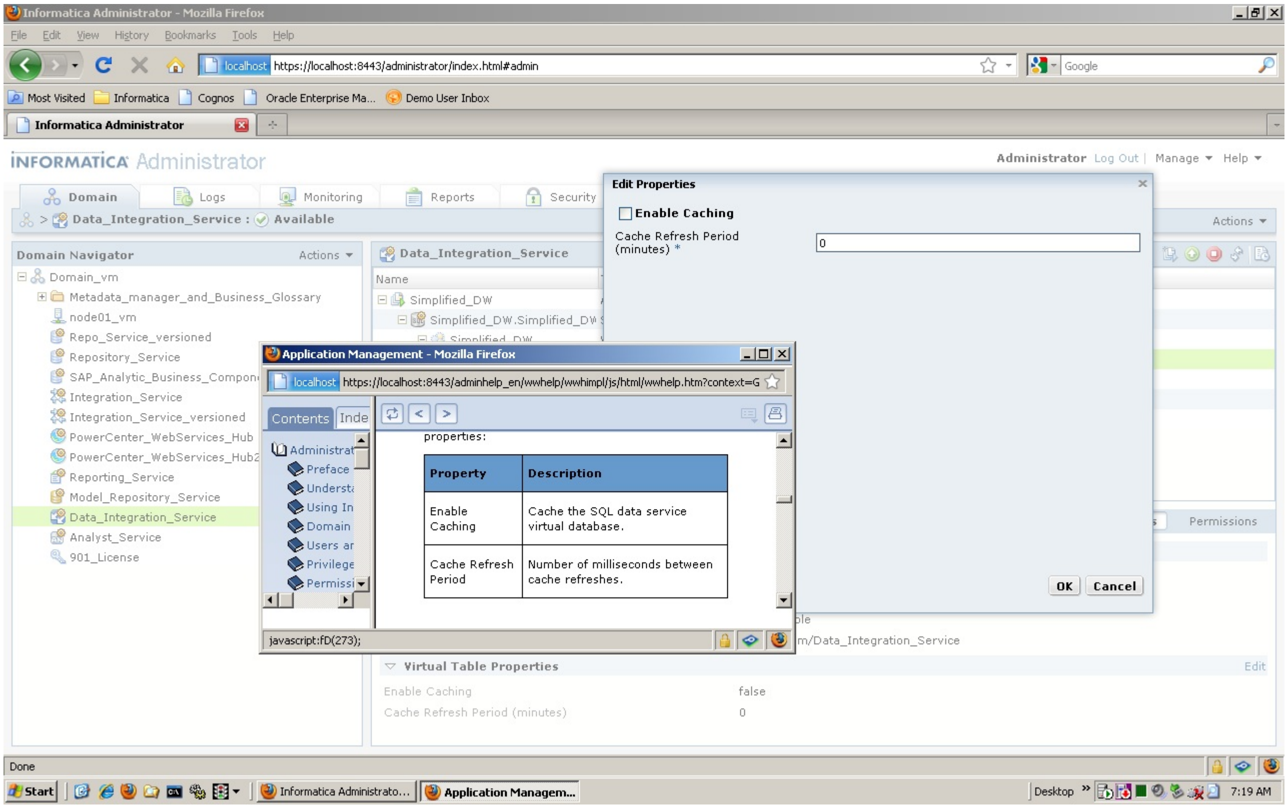


Figure 32 *Virtual table with a cache*

Administrators can determine whether a cache should be defined and how the cache of the virtual table should be refreshed: once or periodically. If periodically, we can specify at what time. Refreshing of the cache can be scheduled or it can be done manually. A cached virtual table is sometimes referred to as a *materialized view*. Defining a cache for a virtual table is simple; see Figure 33.


Informatica Administrator - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://localhost:8443/administrator/index.html#admin
Most Visited
Informatica
Cognos
Oracle Enterprise Ma...
Demo User Inbox
Informatica Administrator
INFORMATICA Administrator
Administrator Log Out | Manage Help
Domain
Logs
Monitoring
Reports
Security
Data_Integration_Service : Available
Actions
Domain Navigator
Domain_vm
Metadata_manager_and_Business_Glossary
node01_vm
Repo_Service_versioned
Repository_Service
SAP_Analytic_Business_Compon
Integration_Service
Integration_Service_versioned
PowerCenter_WebServices_Hub
PowerCenter_WebServices_Hub2
Reporting_Service
Model_Repository_Service
Data_Integration_Service
Analyst_Service
901_License
Name
Simplified_DW
Simplified_DW.Simplified_DW
Application Management - Mozilla Firefox
https://localhost:8443/adminhelp_en/wwhelp/wwhimpl/js/html/wwhelp.htm?context=G
Contents
Inde
Administrat
Preface
Underst
Using In
Domain
Users ar
Privilege
Permissi
javascript:fD(273);
Permissions
m/Data_Integration_Service
Edit
OK
Cancel
Done
Start
Informatica Administrato...
Application Managem...
Desktop
7:19 AM

Edit Properties

□ Enable Caching

Cache Refresh Period (minutes) * 0

properties:

Property	Description
Enable Caching	Cache the SQL data service virtual database.
Cache Refresh Period	Number of milliseconds between cache refreshes.

Virtual Table Properties

Enable Caching false

Cache Refresh Period (minutes) 0

Figure 33 *Caching data for a virtual table*

As indicated, caches can be stored in files or in databases. Retrieving cached data stored in files is normally very fast, especially if large subsets of all the records must be retrieved. If a query on a cached virtual table only needs a small set of rows or if it contains a group-by operation, an index might be useful to speed up that query. In that case, a table is recommended because it can be indexed.

There can be various reasons for defining a cache:

- Load optimization: A cache might be useful to minimize the load on the underlying system. It could be that a virtual table is defined on foreign tables in an old system that already has issues with its performance. Additional queries might be too much for this system. By defining a cache, fewer queries will be executed on the old system.

- Consistent reporting: A cache could also be useful if a user wants to see the same report results when he runs a report several times for a specific period of time (a day, week, or month). This is typically true for users of reports. It can be quite confusing when the same report returns different results. In this case, a cache might be necessary if the contents of the underlying database are constantly being updated.

- Source availability: If the underlying system is not always available, a periodically refreshed cache might enable 7x24 operation.

- Complex transformations: The transformations that must be applied to the data might be so complex that doing them on-demand might be too slow. Storing the transformed result in the cache and reusing the result several times, might be more efficient.

The side effect of caching is that the data returned when querying the virtual table may no longer be 100% up-to-date. Plus, caching means that we switch from on-demand transformation to scheduled transformation.

By defining several cached virtual tables on one and the same virtual table that contains the required data, different users can access the same data even when they need different caching settings. For example, we could define two virtual tables V_1 and V_2 on top of virtual table V_3. V_3 contains the right data for all users. V_1 shows all the data of V_3 and uses a cache that is refreshed once a day, whereas V_3 also shows all the data of V_3 but uses a cache that is refreshed once a month.

24 Optimization Techniques for Accessing Foreign Data

The previous section described caching as a mechanism for optimizing queries. This section describes some of the query optimization techniques supported by Informatica Data Services. It starts with explaining the factors influencing query performance.

The performance of a query executed by Informatica Data Services is determined by three main factors: the time it takes to retrieve data from the foreign tables, the time it takes to transmit the data to Informatica Data Services, and the time it takes Informatica Data Services to process all the mappings and queries.

The database server itself, the first factor, is responsible for optimizing database access, meaning it's responsible for the amount of data transferred from the disks to the database server. For a large part, the amount of data transferred determines the performance of a query. Informatica Data Services has limited influence on this factor, except that it should try to send queries that are easy to optimize for the database server.

The second factor is an important one. Informatica Data Services is responsible for optimizing data traffic between the database server and itself. Again, the amount of data sent over the network between the servers determines for a large part the performance of the queries. This means that Informatica Data Services must optimize the amount of data transferred between the data stores in which those tables reside and itself. The module called the *optimizer* is responsible for this. Informatica has spent a lot of time to research how to optimize this optimizer.

The third factor, the processing of the mappings and queries, is something for which Informatica has build up a lot of experience with their other product PowerCenter.

The Informatica Data Services optimizer is very much comparable to the optimizer of a SQL database server. The difference though is that optimizers of database servers try to optimize the amount of I/O, whereas the optimizer of Informatica Data Services tries to optimize the amount of data traffic between the data stores and itself. Next we explain the key features of the Informatica Data Services optimizer including:

- Combining queries
- SQL pushdown
- Parallel processing
- Distributed joins
- SQL override
- And other advanced query optimization techniques.

Optimization through Combining Queries – First of all, if a query (that results from translating the mapping) is executed on a virtual table defined on a number of other virtual tables, which are also defined on other virtual tables, how does Informatica Data Services execute these queries? What won't happen is that query after query is executed sequentially. Imagine that a query is executed on virtual table V_1 which is defined on virtual tables V_2 and V_3, and those are defined on tables T_2 and T_3 respectively. What Informatica Data Services will not do is, first, execute V_2's query on T_2, then V_3's query on T_3, and next combine and join those results and the result is kept somewhere in memory, and finally the query on V_1 is executed on that intermediate result. Although this approach will return the correct result, the process is slow. The approach taken by Informatica Data Services is that queries are combined into one query that is passed to the database server to be executed. So, in the example the queries belonging to the virtual tables V_1, V_2, and V_3 plus the query entered by the application are combined into one query that leads to a join of tables T_2 and T_3. So it will be the database server that's doing the join and not Informatica Data Services. In short, all the layered queries are coalesced into one single comprehensive query and optimized accordingly. This will definitely minimize traffic between the database server and Informatica Data Services.

Optimization through Pushdown – Another technique for optimization is called *pushdown*. Because, as indicated, the goal of the Informatica Data Services optimizer is to minimize the amount of data it gets back from the foreign data stores, it will try to 'push' as much processing to the underlying database servers themselves. This means that selections (such as, get only the customers from London), projections (such as, get only the names and addresses of the customers), and group-by operations are pushed down. Informatica Data Services will let the database server process them. Evidently, this is not possible for every data store. For example, if the data store is a sequential file, an XML document, or a SOAP service, no optimization can be executed by the data store, Informatica Data Services has to do all the work instead. It will retrieve all the data (unless a cache exists because then the data is retrieved from that cache).

Optimization through Distributed Joins – If a query is a one-table query, it's entirely pushed down to the database server and only its result is returned. If two tables are joined and both are stored in the same database, again a query with a join is pushed down to the database server. It's a little bit more complex if two tables are joined and if they are not managed by the same database server. An inefficient strategy would be to retrieve all the data from both database servers and let Informatica Data Services do the join because it would involve a lot of data traffic.

Informatica Data Services supports a few techniques to process this more efficiently, which we will discuss next.

If one of the two tables is relatively small (approximately 100,000 rows or less), that table could be read first and kept in memory. Next, the large table is read, and row by row comparisons are made with the smaller table in memory.

When two large tables must be joined, Informatica Data Services will use a so-called *sort-merge join*. In this case, both tables are retrieved from the two data stores and both have been sorted by the database server on the join columns. The effect is that the database servers perform the sorts, and Informatica Data Services only has to do the merge. Let's take the following join as an example:

```
SELECT   TL1.COL1, TL1.COL2, TL2.COL1, TL2.COL2
FROM     TABLE_LARGE1 AS TL1 INNER JOIN TABLE_LARGE2 AS TL2
         ON TL1.COL1 = TL2.COL1
WHERE    TL1.COL3 > 1000
```

The following statement is send to the first data store:

```
SELECT   COL1, COL2
FROM     TABLE_LARGE1
WHERE    COL3 > 1000
ORDER BY COL1
```

And a comparable statement is send to the other:

```
SELECT   COL1, COL2
FROM     TABLE_LARGE2
ORDER BY COL1
```

Merging the two results is straightforward for Informatica Data Services. The advantage of this approach is that the work still to be done is the merge-step and that should not take that much processing time. Still, a lot of data is sent, but most of that processing is done in parallel.

When tables are joined and one or more are cached, it might be that the cache is used. Imagine that tables T_1, T_2, and T_3 are joined together and that a cached virtual table is available on the join result of T_2 with T_3. In this case, Informatica Data Services will perform a join of T_1 using the cache. If that cache is small, Informatica Data Services might use the distributed semi-join approach again to join them.

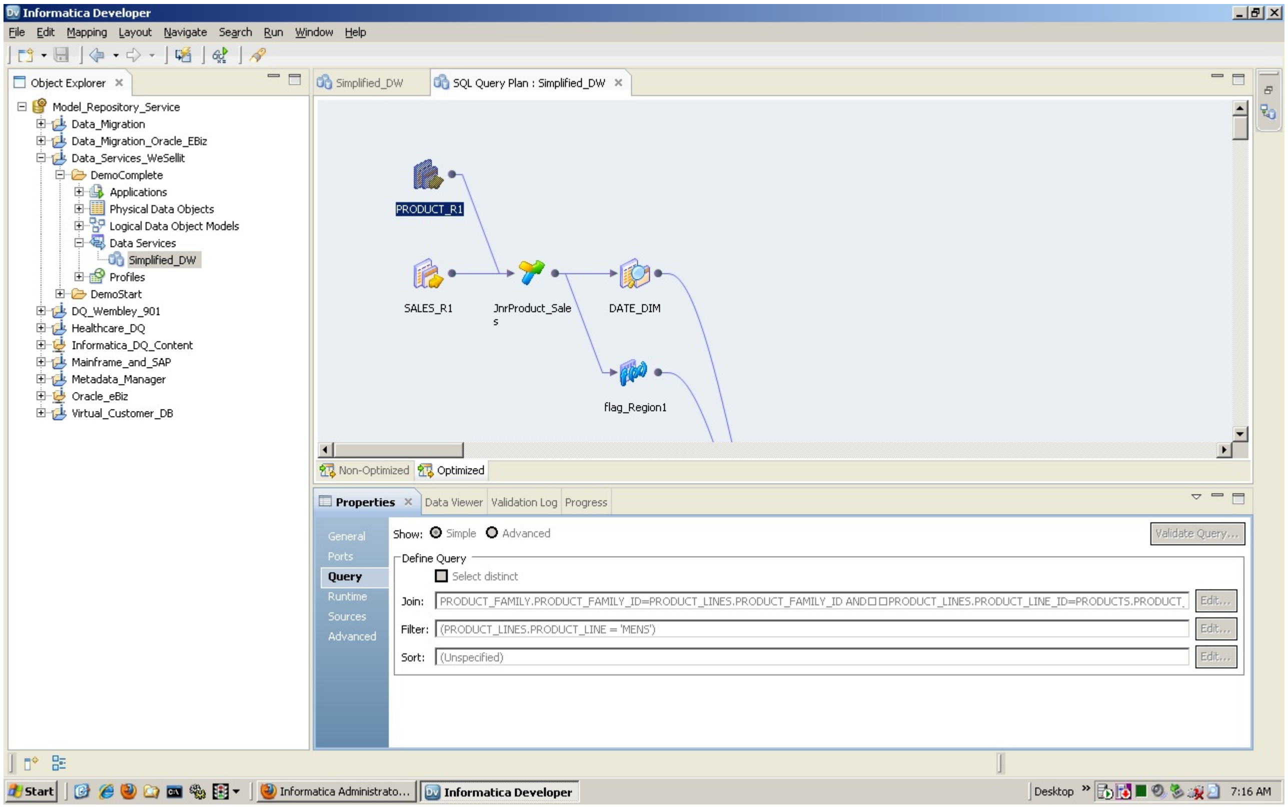
Another form of optimization is that as much work as possible is 'pushed down' to the underlying database server(s). For example, if a join has to be executed between tables T_1, T_2, T_3, and T_4, and if the last three of those tables are stored in the same database, a query that contains a join of those three tables is send to the database server. Then the result of this join is joined with table T_1. Push down is an optimization mechanism that aims at doing as much of the processing as possible as close to the data. Plus, if more database servers are involved, processing can be done in parallel.

And if nothing else works, Informatica Data Services performs a nested loop join, which is not fast, but will return a correct result.

For most of the decisions the Informatica Data Services optimizer has to make, it must know the approximate size of the result set on each side of the join. Informatica Data Services automatically gathers such data from the data stores. This type of data is sometimes called *statistical data*. Of course there are sources that can't offer that type of data. For example, XML documents and SOAP services can't answer questions concerning the number of rows they contain. In that case, statistical data can be entered by hand.

If developers feel that the optimizer isn't coming up with the best possible strategy *commands* and *hints* can be specified. With commands the optimizer is forced to use a specific strategy and with hints the optimizer can follow any strategy.

Another optimization form is that developers can see what the processing strategy of Informatica Data Services for a query will be. They can see the order in which tables are joined; see Figure 34. If they can come up with a better strategy, they can change the order in which particular operations take place.



Show: ● Simple ○ Advanced

Define Query

□ Select distinct

Join: PRODUCT_FAMILY.PRODUCT_FAMILY_ID=PRODUCT_LINES.PRODUCT_FAMILY_ID AND□□PRODUCT_LINES.PRODUCT_LINE_ID=PRODUCTS.PRODUCT_

Filter: (PRODUCT_LINES.PRODUCT_LINE = 'MENS')

Sort: (Unspecified)

Figure 34 *An example of a query execution plan*

SQL Override – Developers can choose to override the generated SQL query with their own custom SQL. In those queries they can use the full power of SQL, except that, although they're allowed to join data from different tables, they can't join tables from different data sources.

To summarize, Informatica Data Services supports many approaches and techniques to optimize access to the data stores. Together, they give developers and administrators various instruments to optimize the performance of queries, and to influence when and how queries are executed. Research in this area will continue and must continue.

25 Security Features

Informatica Data Services offers a rich set of security features, including authentication, authorization, and encryption. All three are described in this section.

With respect to *authentication*, users who access Informatica Data Services must present credentials (such as user id and password) to identify themselves; in other words, Informatica Data Services checks whether the user is really who he says he is. Informatica Data Services can be configured in such a way that an external system is used for authentication. In that case Informatica Data Services will ask the external system to perform the authentication.

Users can be introduced and defined within Informatica Data Services, but user definitions can also be stored outside Informatica Data Services, for example in LDAP directories. Users can be grouped in domains and in user groups. Unfortunately, groups can't be nested.

It's not likely that every user should have access to all the data accessible through Informatica Data Services. Therefore, Informatica Data Services offers features for *authorization* to control which user is allowed to access which data elements.

To each user group and individual user access privileges for virtual tables and data services can be assigned. This is somewhat similar to assigning privileges to users with the GRANT statement in SQL. The following types of privileges are supported: read, write, execute, select, update, insert, and grant.

If users enable pass-thru, which means SQL statements are passed straight-on to the underlying database server without any changes by Informatica Data Services, besides the privileges in Informatica Data Services, the user must have the privileges inside those underlying database servers to access certain tables.

It might be that two users may query the same virtual table, but that they are not allowed to see the same rows. For example, a manager might see the data of all the customers, but an account manager may only see the customers for whom he is responsible. Therefore, Informatica Data Services supports *row-level security*. In the definition of a virtual table we can add a condition that includes the id of the user so that only those rows that the user is allowed to access are returned.

If applications call the virtual tables through a SOAP service interface, there might be a need to encrypt those calls. Informatica Data Services accomplishes *encryption* by using SSL as the connection mechanism.

26 Inserting, Updating, and Deleting Data

In the current release, the virtual tables and physical data objects in Informatica Data Services are read-only. Virtual tables can be created that can be inserted, update, and deleted, but only if they are developed using virtual stored procedures. The next release will support more functionality with respect to updateability.

27 Technical Advantages of Informatica Data Services

This section lists the technical advantages of Informatica Data Services. The next section describes its business advantages.

***On-Demand Data Integration* –** Data doesn't have to be stored in separate, derived data stores before it becomes accessible for reporting and analytical tools. When those tools access the data via Informatica Data Services it's retrieved from the data stores live, it's integrated, transformed, and cleansed live without the delay of copying the data to the data stores.

***Access of Non-Relational Data Stores* –** Not all the data users need is stored in SQL database servers. Therefore, Informatica Data Services allows data to be stored in and accessed from flat files, XML documents, SOAP-based services, and spreadsheets. Data from all these different data sources can be merged to create an integrated result.

***Seamless Switching between On-Demand and Scheduled Transformations* –** If the need exists to store integrated data in a data store instead of doing it on-demand, the virtual tables or mappings can be reused or physically materialized for that purpose.

***Wide Range of Transformation Operations* –** The mapping language with which developers indicate how tables should be transformed supports a wide range of transformation operations, ranging from simple string manipulation operations to complex data cleansing operations, such as de-duplication.

***Easier Data Store Migration* –** Data store independency means that a report that accesses a particular data store can easily be migrated to another data store. Informatica Data Services can redirect the reports' queries to another data store. For example, when a report is currently accessing a data mart, migrating it to the data warehouse doesn't require any changes in the report definition. The same applies when it's necessary to migrate from files to SQL database technology, or when a product, such as SQL Server, must replaced by Netezza. In most cases, these changes will have no impact on the reports. In short, if Informatica Data Services is in place, migration to another data store (technology) is easy. There might be various reasons why an organization wants to migrate. For example, they may want to use technology that offers faster query performance, or they have outsourced their data storage and this needs to be accessed differently.

Shareable Transformation Specifications – By stacking virtual tables, common specifications relevant to multiple applications can be implemented in bottom-level virtual tables and the more application-specific specifications can be implemented in the top-level tables. The advantage is that tools will share common specifications which will lead to more consistent results, and it will improve development and maintenance time.

Advanced Security Rules for Data Access – Independent of all the security rules implemented in the underlying data stores, in Informatica Data Services a central security layer can be defined. Users don't have to log on to the underlying stores, they only have to log on to Informatica Data Services. Informatica Data Services supports authentication, authorization, and encryption.

Query optimization techniques – Informatica Data Services has implemented several techniques for improving query performance. Most of those techniques try to minimize the amount of data to be transmitted between the federation server and all the data stores. This is quite important because in an on-demand integration environment, users are waiting for the results.

Advanced Caching Mechanism – The caching mechanism can be used to minimize interference on the data stores; it can be used to avoid that complex transformations and cleansing operations are executed repeatedly; and it can be used to present consistent results over a certain period of time.

Lineage and Impact Analysis – Relationships between all the mappings, physical data objects, and virtual tables are stored in a central repository. This allows Informatica Data Services to show all those relationships.

On-Demand Data Profiling – Data profiling has been fully integrated with Informatica Data Services. When defining mappings and virtual tables it becomes possible to use profiling functionality to verify whether all the required transformation rules have been implemented. In addition, users can be invited to participate in this process. This could improve the quality of the process.

On-Demand Data Cleansing – Most data federation servers are only capable of executing simple cleansing operations. By implementing cleansing rules in the mappings Informatica Data Services is able to perform the most complex cleansing rules in an on-demand fashion.

Transparent Archiving of Data – Eventually data warehouses become so big that 'older' data has to be archived. Data is normally archived to a cheaper storage mechanism that can still be accessed. But if data is old, it doesn't always mean that no users are interested in it anymore. Informatica Data Services can hide where and how archived data is stored. The fact that the data has been archived can be hidden for the data consumers. If users are still interested in all the data, Informatica Data Services can combine the non-archived data with the archived data. The effect might be that the performance is (somewhat) slower, but report specifications don't have to be changed. Thus, Informatica Data Services hides that some data has been archived.

28 Business Advantages of Informatica Data Services

Informatica Data Services offers the following business advantages:

Cost Reduction due to Simplification of Business Intelligence Architectures – If Informatica Data Services is installed in an existing business intelligence architecture, for example, one based around a central data warehouse, Informatica Data Services makes it possible to simplify the architecture. Data marts and cubes can be removed and the existing reports must be redirected to another data store, which, as indicated, is easy to do with Informatica Data Services. The advantage of this simplification of the architecture is cost reduction.

Self-service Reporting and Analytics – If organizations allow their users to create their own reports and do their own analysis, the underlying architecture has to be flexible. It must be possible to react quickly to new demands of users. If new tables must be created or existing ones modified, it must be easy. Informatica Data Services supports this due to the fact that all the virtual tables are really virtual. Adding new ones or changing existing ones does not involve a complex process where table are created and loaded. A virtual solution is more flexible than a stored solution.

Fast Handling of Changing User Needs – If user needs change, if they want to change existing reports, there is no need to change underlying data store structures. The business impact will be considerable, because users can get reports in days instead of weeks or maybe even months.

Decreased Time-to-Market – For developing new reports it's not required to first design and develop a data store-based solution. If the data needed is stored somewhere in one of the data stores accessible for Informatica Data Services, it can be transformed in the way the report requires. There is no need for introducing new data stores or changing existing data stores. This improves the time it takes to develop the reports. In addition, because most specifications already exist within Informatica Data Services and can be re-used, it takes less time to develop a new report. Development can focus primarily on the use of the specifications.

Operational Reporting and Analytics – If Informatica Data Services has access to operational databases, data consumers can run reports on the operational data and see the most recent state of the data. Reports can also be developed in which current data from an operational database is joined with historical data from a data warehouse. Care should be taken that accessing operational databases might lead to too much interference on those operational systems. Caching might be required to minimize the query workload and therefore the interference.

Collaborative Data Profiling and Cleansing – Informatica Data Services makes it possible that developers and analysts work together so that data is transformed correctly and that the right data quality level is reached.

Increased Trust in Data – Because business users are involved in data profiling and cleansing process, it will increase the trust they will have in the reported data they use in their decision making process.

Consistent Reporting – With Informatica Data Services all reporting and analytical tools used by an organization can share the same transformation specifications. Therefore, the results the users view will be consistent, even if the tools are from different vendors. This improves the perceived quality of and trust in the entire business intelligence environment.

Seamless Adoption of New Technology – New database and storage technology keeps appearing on the market, such as data warehouse appliances, analytical databases, columnar databases, and solid state disk technology. As indicated, because Informatica Data Services separates the data consumers from the data stores, replacing an existing data store technology with a new one is relatively easy and has no impact on the reports. This makes it easier for organizations to migrate to better, cheaper, more powerful database technology.

29 Case Studies

This section contains two brief case studies of organizations using Informatica Data Services. Note that both organizations use the virtualization capabilities of Informatica Data Services not only for business intelligence purposes but also for operational applications.

HealthNow New York – HealthNow New York is one of New York's leading healthcare companies, but not one that simply pays bills. It's a proactive partner in health, providing access to outstanding healthcare, innovative technologies, and information and services that help people lead healthier lives every day. HealthNow serves more than 815,000 members, 13,000 client companies, and 2,100 employees. In 2008, company revenues grew to more than $2.27 billion.

With an immense amount of diverse data spread throughout the organization and no unified data integration solution, HealthNow New York was struggling to adapt to the changing climate of the healthcare industry. At a business level, the organization needed to supply a single common framework for access to enterprise-wide healthcare data, including member information, drug claims, provider data, laboratory results, and encounter details. Data was spread across up to 16 enterprise databases—including DB2, Sybase, and SQL Server—and more than 30,000 Microsoft Access databases. It took HealthNow New York months to build critical new data extracts of customer and products data for pricing and risk analysis. And it required 1,700 hours to add a new product to the portfolio.

HealthNow New York decided they needed a different solution, one based on virtual data integration instead of physical data integration where data is copied from one data store to another. Virtual data integration, or data virtualization, enables the on-demand integration and access of multiple and diverse data sources. They selected Information Data Services to be the foundation of their integration strategy and build a data virtualization architecture; see Figure 35.

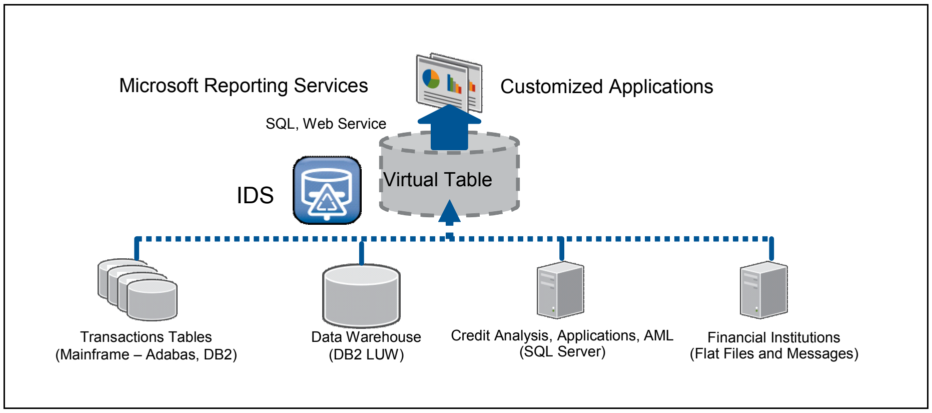


Figure 35 *HealthNow's data virtualization architecture*

Currently, HealthNow New York processes 30,000 claims on average a day through the Informatica Platform. The company has developed approximately 4,000 workflows, and by the end of 2010, HealthNow New York will support nearly 10,000 unique requests and will have up to 100 concurrent users. In terms of response time, HealthNow New York is able to rapidly process more than 10 million rows from Sybase/SQL Server to cache 100,000 rows with a sub-second response time for Web services.

HealthNow New York is experiencing the following advantages:

- **Increased responsiveness to business needs.** Where it previously took 1,700 hours to add a new product to the portfolio due to system and data complexity, today HealthNow New York can build a domain model around core areas such as claims so that the application layer can access data through data services. Instead of taking weeks to get the appropriate data feed ready, it can be done in hours.
- **Single version of the truth.** Previously, the business had to go to five different places to get claims information. With the Informatica Platform, HealthNow New York can have a data service for data feeds such as "paid claims" and "pending claims," and business users have to go to only one place though a Web-based tool to access this data.
- **Reduction in ancillary data stores.** Lacking easy access to data, HealthNow New York business units created lots of custom Access databases; this customization led to inconsistent information and the risk of exposing personal health information. Informatica technology enables HealthNow New York to easily access data, analyze the data, and then quickly provision data services for any application.
- **Decoupling business from data in support for the SOA.** If a data source changes, HealthNow New York just changes which data source the data services abstraction layer points to without affecting the consuming application. Standards-based data abstraction has led to a threefold reduction in the costs associated with maintaining data feeds.

HealthNow New York is constantly examining other uses for the Informatica Platform and Informatica Data Services to help increase efficiency, reduce redundancy, shorten time to market and improve overall quality. To simplify testing in its data warehouse environment, for example, the company has federated the warehouse and the data marts, allowing testers to easily compare data within a single environment, without having to export the data and then do a comparison. This will reduce test development and execution time by up to 50 percent. HealthNow New York is also simplifying what the underlying sources of data look like. For the application developer, HealthNow New York exposes one data service that combines five sources of data from many various databases. This consolidation will reduce development time by up to 25 percent; while there will be a corresponding 75 percent reduction in redundant data stores.

Large Brazilian Bank – A large Brazilian bank had a number of challenges:

- A lack of visibility for proper supervision and regulation of the national financial system.
- A need for real-time analysis and joining of data stored in a wide range of database servers, including Software AG Adabas, IBM DB2, and Microsoft SQL Server. In addition, huge amounts of data was stored in flat and index-sequential files.
- A need for persistent data replication even for one-time use data.
- Handling of huge data volumes. The production environment consisted of 6 Terabytes and the data warehouse environment of 14 Terabytes.
- Users of different tools across a heterogeneous set of data sources.

The decision was made to look into virtualizing the various data sources using Informatica Data Services; see Figure 36.

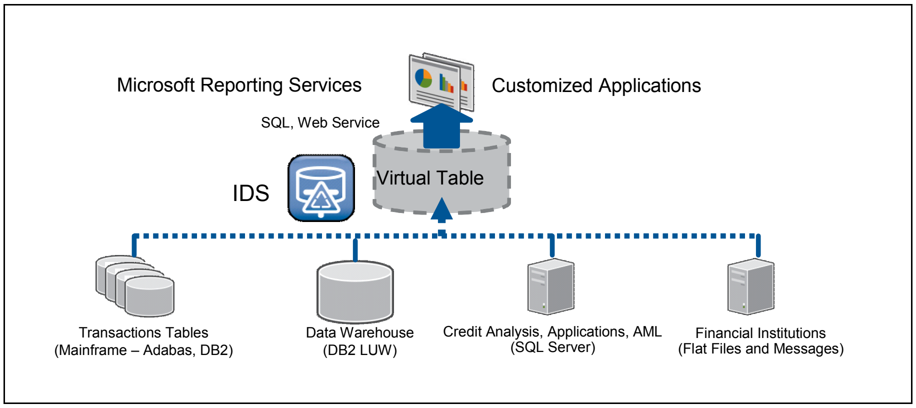


Figure 36 *Virtualizing the data stores*

In Informatica Data Services logical data models can be created that represent core business entities. The data for those core entities is spread across different platforms and different database servers. By using mappings the data can be joined, transformed, and cleansed to fit the logical data definitions. In addition, logical data objects can be created to link to PowerCenter, a product they had been using and were familiar with.

This virtualization offers numerous benefits. First of all, speed of delivery. New virtual tables can be created in less than one hour. Secondly, risk/fraud governance can be improved across more than 6,000 financial institutions, and they can implement BASEL I, BASEL II, and SOX compliancy. Thirdly, they can present a single view of the truth. Users can access consistent customer and plan rate data. And the fourth advantage is that they will be able to manage and administer the logical data objects centrally.

About the Author Rick F. van der Lans

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

Rick is chairman of the annual European Data Warehouse and Business Intelligence Conference (organized in London), chairman of the BI event[9] in The Netherlands, and he writes for the B-eye-Network[10]. He introduced the Data Delivery Platform in 2009 in a number of articles[11] that were published on BeyeNetwork.com.

He has written several books on SQL. His popular *Introduction to SQL*[12] was the first English book on the market in 1987 devoted entirely to SQL. After more than twenty years, this book is still being sold, and has been translated in several languages, including Chinese, German, and Italian.

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

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.

[9] See http://www.bi-event.nl/59857

[10] See http://www.b-eye-network.com/channels/5087/articles/

[11] See http://www.b-eye-network.com/channels/5087/view/12495

[12] 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