it-novum

Open Business Solutions. Delivered.

Best Practice

Open Data Warehouse Building a Data Warehouse with Pentaho

Contents

1. Objectives and benefits

This best practice guide shows how a powerful, high-performance data warehouse system can be built with open source products. This best practice guide is of interest to you if you

- are thinking about introducing a data warehouse system into a company and are looking for an appropriate solution
- are more interested in software adaptation than investing your money in software licensing
- want an open and customizable solution that can quickly be put in place and will grow with your needs
- already have open source solutions in mind, but are unsure how to approach such a project
- have a limited project budget

You will benefit from reading this best practice guide, because it

- clearly demonstrates and describes an example architecture for an open source-based data warehouse
- gives you tips for designing and implementing your system
- provides a schema that you can modify to suit your own individual needs
- contains recommendations for proven software products

2. Management Summary

Companies and the environments in which they exist are producing ever faster and ever greater amounts of data. The resulting data contains an enormous economic potential – but only when it has been properly processed and analyzed. A data warehouse solution provides the ideal basis for this processing and analysis. Setting up and developing a data warehouse, however, often entails costly licensing and hardware issues. This deters mainly small and medium-sized companies from undertaking such a project, even though cost-effective open source solutions have long been available in the business intelligence sector. Are these solutions suitable, however, for developing a high-performance data warehouse system or is there really no alternative to commercial software products when it comes to performance and functionality?

This best practice guide addresses whether and how a powerful data warehouse can be built using open source systems.

For this purpose, we will be designing and building a prototypical example architecture using the Pentaho and Infobright solutions. We have implemented this architecture – with its essential features – in a number of projects, many of which have been in successful use for years.

For ease of understanding, this best practice guide has been organized in the following manner:

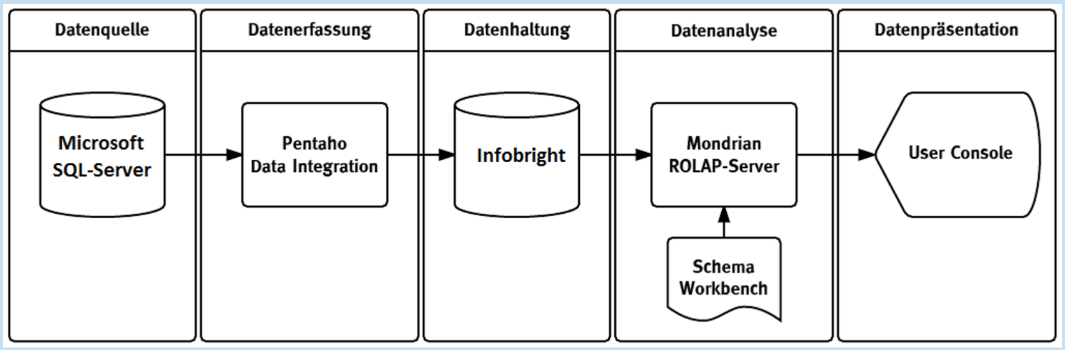
- General construction of a data warehouse
- Connecting data sources
- Designing the ETL process
- Data storage & management
- Data analysis
- Data presentation
- Conclusions

In the process, all pertinent points associated with the design and implementation of a data warehouse will be covered.

3. Building a data warehouse system

A data warehouse system consists of five levels: Data sources, data acquisition, data management, data analysis and data presentation. The individual layers are realized by means of various concepts and tools and their respective implementations are based on the requirements of the given company.

The data warehouse system used for our prototype is structured as shown in the figure below. All components are open source. The Microsoft database AdventureWorks will serve as the data source. The ETL process at the data acquisition level will be realized with Pentaho Data Integration. This tool is part of the open source solution Pentaho Business Analytics Suite, which, in our example, will be used for data analysis and presentation.



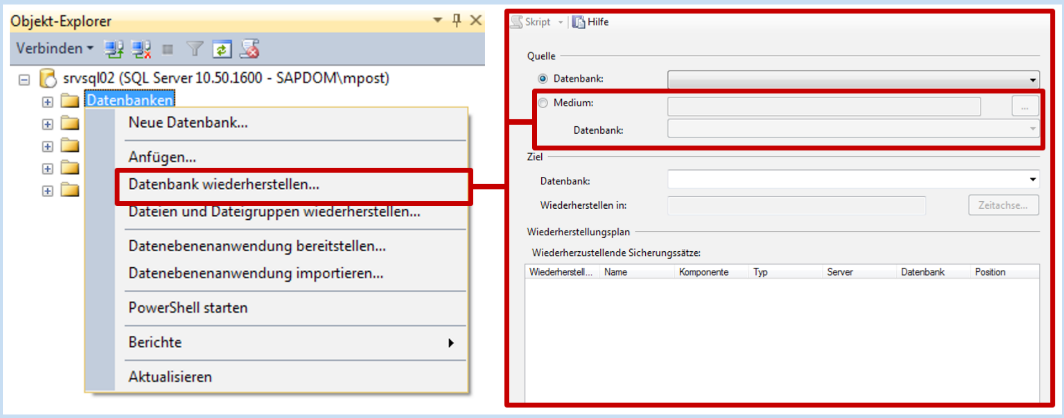
Structure of the sample architecture

The actual data warehouse used within the data management system will be the analytical database management system, Infobright. Pentaho Mondrian will be used as the OLAP server. The required XML schema will be generated with Pentaho Schema Workbench. Once these components have been implemented, the data at the data presentation level can be prepared in the form of analyses, dashboards, and reports using a variety of Pentaho tools. In the following chapters, the individual layers and their associated tools will be described in detail.

4. Data sources

In our prototype system, we will be using AdventureWorks2008R2 (AWR2) as the basis for our sample data. This OLTP database from Microsoft can be downloaded as a free backup file here: http://msftdbprodsamples.codeplex.com/ The data represents a fictitious, multinational company in the production sector that specializes in the manufacture and sale of bicycles and related accessories. The data model consists of more than 70 tables divided into five categories: Human Resources, Person, Production, Purchasing, and Sales.

The data model is similar to the database structures of real companies in terms of scenarios and complexity of database structure and is, therefore, well suited for demonstration and testing purposes. The backup file (.bak) provided can be easily integrated into Microsoft SQL Server. To do this, open SQL Server Management Studio and, clicking on the database symbol, open the restore database interface (see illustration). The new database will then appear in the Object Explorer.



Import the backup file

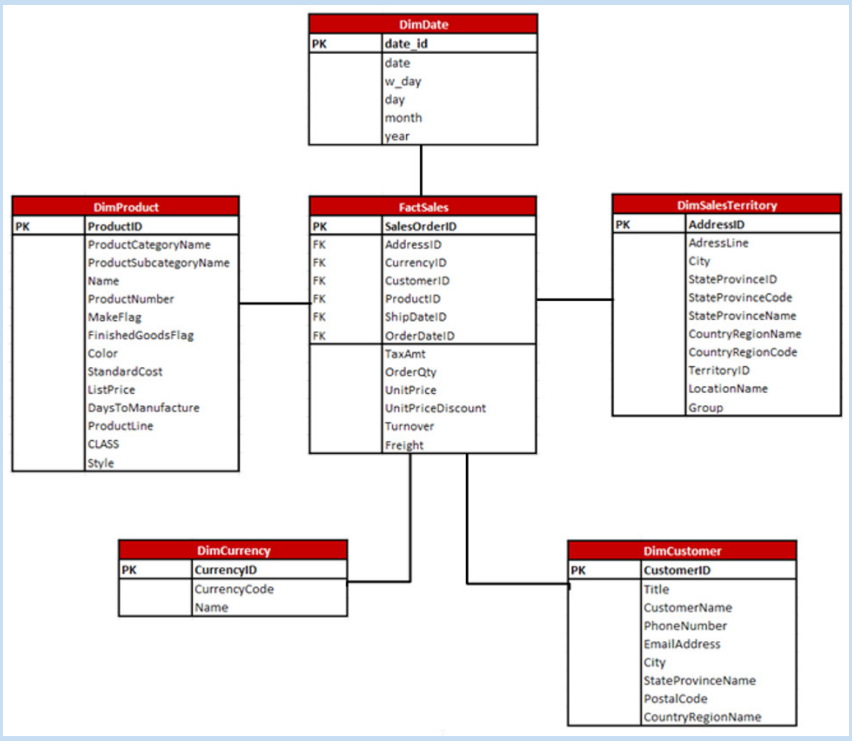
Microsoft SQL Server is not, of course, open source software. We have, however, employed it for this example so that we can use the test data supplied by Microsoft. Alternatively, the backup can be converted into CSV files. These files can then be loaded into any open source database via the Bulk Load option.

5. Data capture

Data acquisition is concerned with the extraction, transformation and loading (ETL) of operational data into the data warehouse. The goal is to achieve a uniform, high-quality base of data in the data warehouse that can be analyzed as needed. Planning the data model is also one of the tasks associated with data capture.

The data model then forms the basis for the later transformation of the data. Certain fundamental decisions have to be made concerning the business process to be analyzed, the level of detail for the analysis as well as any possible dimensions and values.

We will be evaluating sales data as part of the present prototype. No changes will be made in the level of detail for the data, so all hierarchy levels will be retained. Overall, five dimensions are to be created with respect to the OLTP database. These are date, customer, product, sales territory, and currency. As for the metrics, it should be possible to analyze quantity, shipping costs, discounts granted, taxes, unit costs and revenue. The data will be presented in the form of a star schema, because this is the best modeling variant for achieving our goals (see figure). It is also possible, however, to use a snowflake or flat schema.



Star schema based on the AWR2 database

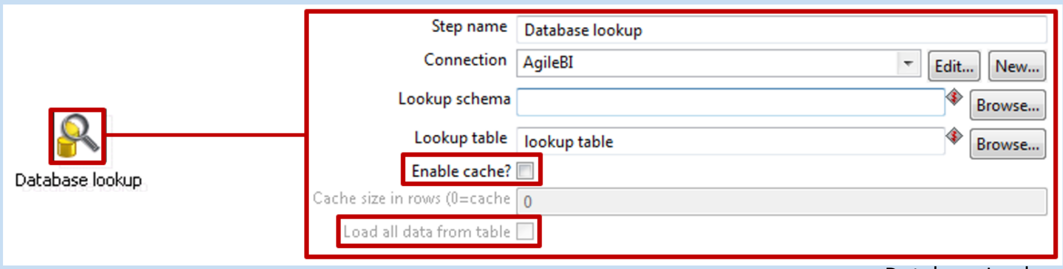
5.1 Designing the ETL process

For the implementation of the ETL process, we will use the open source tool Pentaho Data Integration (PDI). This tool provides predefined steps for extracting, transforming, and loading data. These can be created via a graphical drag-and-drop interface called Spoon. Using Spoon, professional ETL routines in the form of transformations and parent jobs can be built with very little effort. Thus, the amount of manual programming required is very small, making this tool ideal for the novice user.

5.1.1 Best practice approach to working with the Lookup and Join operations

The dimensions for the star schema require information from different tables in the OLTP database. To be able to compare and combine the data, Pentaho Data Integration offers Lookup and Join operations. Each step can be changed or altered, so it is important they are handled correctly. Applying them in the wrong context can lead to performance problems. Since, as a rule, many tables must often be combined within the ETL process, choosing the right steps will significantly increase processing speed. We will, therefore, explain the most important steps below.

Database Lookup
Database Lookup enables the comparison of data from different tables in a database, which allows for the integration of new attributes into the existing database table. Database Lookup was designed for small to medium-sized data sets. Particular attention needs to be paid to how the cache is configured here, because using an active cache can significantly shorten the processing time. A further increase in performance can also be achieved by selecting the checkbox „Load All Data From Table". Particularly with larger data sets, caching can, however, overload the Java stack space causing the transformation to collapse.

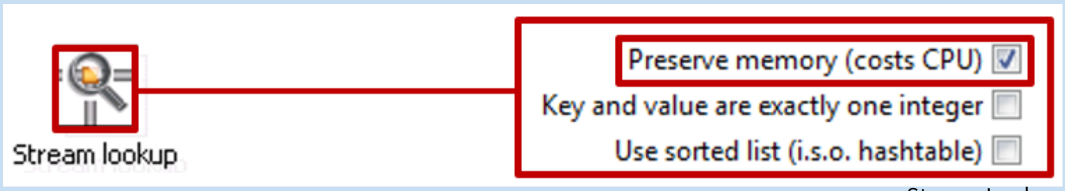


Step name: Database lookup
Connection: AgileBI
Lookup schema:
Lookup table: lookup table
Enable cache? ☐
Cache size in rows (0=cache: 0
Load all data from table ☐

Database Lookup

Stream Lookup

Stream Lookup is similar to Database Lookup with active caching, but the data being compared can be extracted directly from the stream. Due to continuous caching, Stream Lookup is very resource intensive and should only be used with smaller data sets that cannot be loaded directly from a database. By selecting the check box „Preserve Memory (Costs CPU)“, it is possible to reduce the memory footprint, but at the expense of the CPU. When this function is activated, the loaded data is encoded during sorting (hashing). A disadvantage of this, however, is longer running times.

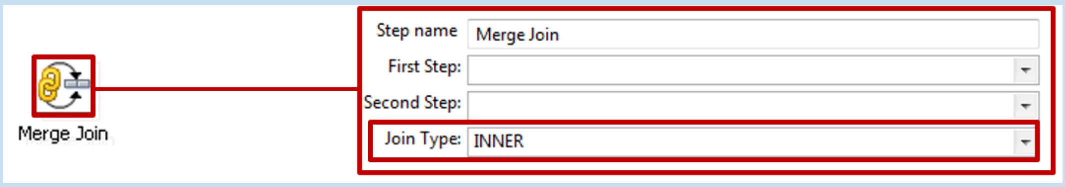


Preserve memory (costs CPU) [✓]
Key and value are exactly one integer [ ]
Use sorted list (i.s.o. hashtable) [ ]

Stream Lookup

Merge Join

Merge Join carries out a traditional join operation within the data integration component. A distinction has to be made here between Inner, Full Outer, Left Outer and Right Outer Joins. Merge Join is particularly suitable for large data sets with high cardinality, that is, many different attribute values within a column. From the beginning, the data must be sorted according to the attributes being compared. Whenever possible, this should be performed directly on the database via „Order By“. Alternatively, this can also be done via Sort Rows, however, this will impact on performance.

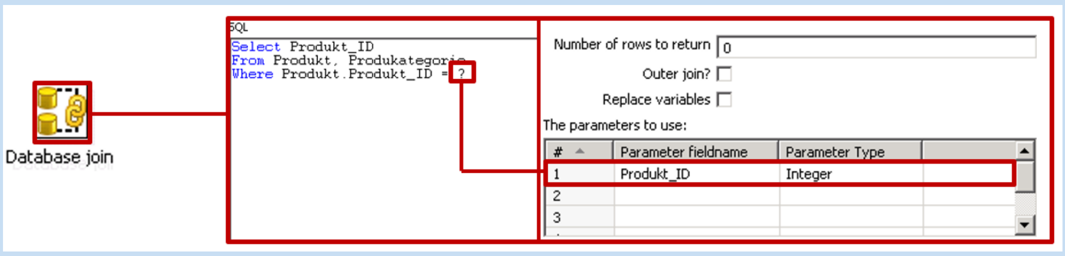


Step name: Merge Join
First Step:
Second Step:
Join Type: INNER

Stream Lookup

Database Join

Using Database Join it is also possible to link database tables and create and execute native queries. The parameters used in the previous transformation can be integrated into the native query. A question mark is used as a placeholder and is replaced by the corresponding parameter when the query is run. When integrating multiple parameters, it is crucial that these be placed in their correct order in the „The parameters to use“ area.



SQL
```
Select Produkt_ID
From Produkt, Produkategori
Where Produkt.Produkt_ID = ?
```

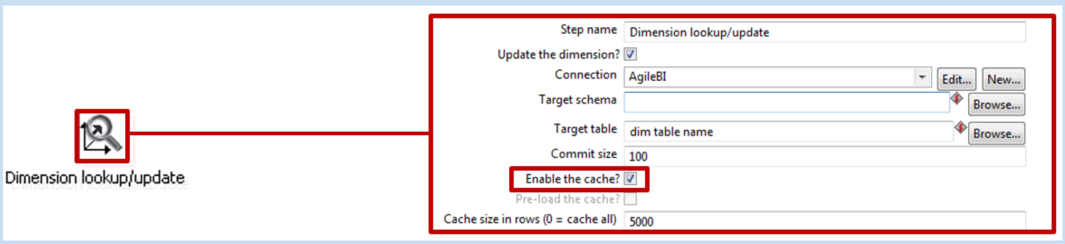
Number of rows to return: 0
Outer join? ☐
Replace variables ☐
The parameters to use:

#	Parameter fieldname	Parameter Type
1	Produkt_ID	Integer
2		
3		

Database Join

Dimension Lookup / Update

The Dimension Lookup / Update step combines the functionality of Insert / Update with that of Database Lookup. It is possible to switch between both functions by selecting or deselecting the checkbox „Update The Dimension“. In addition, this step allows for the efficient integration of the Slowly Caching dimension. The Lookup / Update dimension was developed for special application scenarios and is, therefore, very processor intensive. We mention it here only in the interests of completeness.

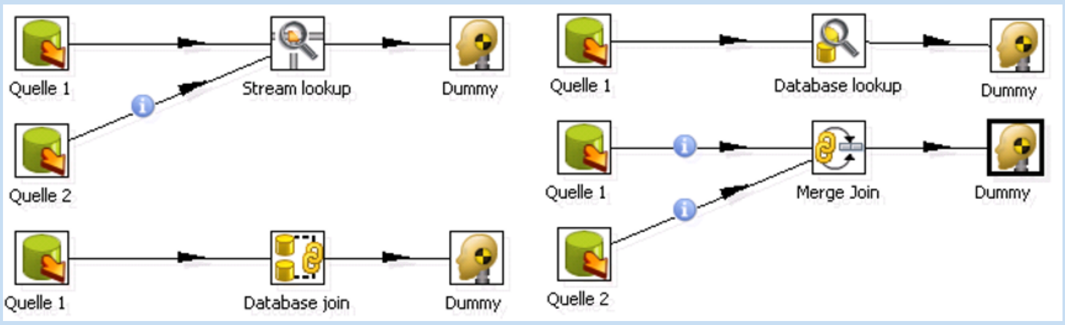


Step name: Dimension lookup/update
Update the dimension? ☑
Connection: AgileBI
Target schema:
Target table: dim table name
Commit size: 100
Enable the cache? ☑
Pre-load the cache? ☐
Cache size in rows (0 = cache all): 5000

Dimension Lookup / Update

Performance Test

To highlight the differences in performance, we have examined the various steps in connection with two data sets that correspond to the expected amounts of data in our demo.



Transformations of the Performance Test

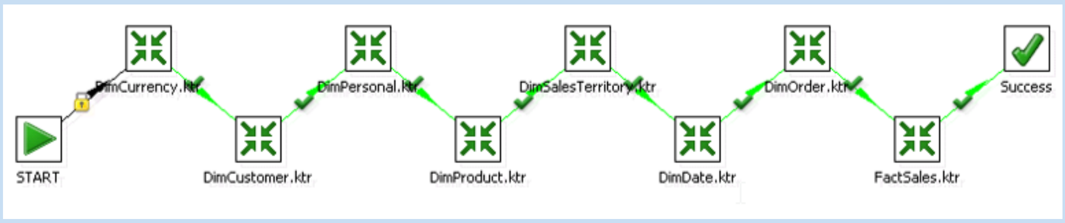
As demonstrated in the table, Database Join highlights major performance problems when increasing data volumes are involved. For the remaining steps, however, there is hardly any increase in processing time despite the significantly larger amounts of data.

Operation	40.000 Data Sets	152.500 Data Sets
Database Lookup	0,9 sec.	1,5 sec.
Stream Lookup	1,6 sec.	2,3 sec.
Merge Join	1,3 sec.	2,6 sec.
Database Join	8,2 sec.	3 min.

Performance Test Results

5.2 Practical implementation

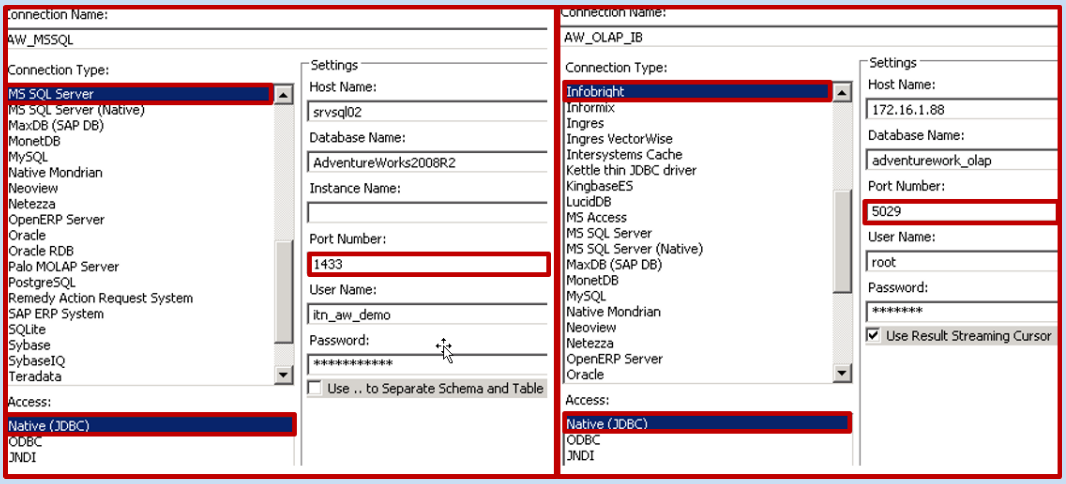
The ETL processes conform to the specifications laid down for a multi-dimensional data model. A transformation is created per dimension or fact table. This provides a better overview and errors can be more easily located. In addition, the risk of overloading the server is also reduced. The various transformations are combined and controlled via a central job:



Parent Job

Before the actual transformation processes for preparing and populating the data warehouse can begin, the database connections to the data source and the data warehouse must first be defined. New database connections can be created in Pentaho Data Integration under the tab View>Database Connections. The selection of the correct database and driver as well as user-specific settings, such as database name or user, is decisive.

It is important to ensure that the JDBC drivers for both databases are in the lib directory of the Data Integration Tool. By default, MySQL can be accessed via port 1433 and the Infobright database via port 5029.

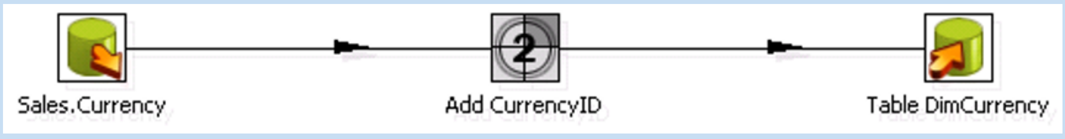

Connection Name: AW_MSSQL
Connection Type: MS SQL Server
Access: Native (JDBC)
Settings
Host Name: srvsql02
Database Name: AdventureWorks2008R2
Instance Name:
Port Number: 1433
User Name: itn_aw_demo
Password: ************
☐ Use .. to Separate Schema and Table

Connection Name: AW_OLAP_IB
Connection Type: Infobright
Access: Native (JDBC)
Settings
Host Name: 172.16.1.88
Database Name: adventurework_olap
Port Number: 5029
User Name: root
Password: *******
☑ Use Result Streaming Cursor

Database connection details

DimCurrency

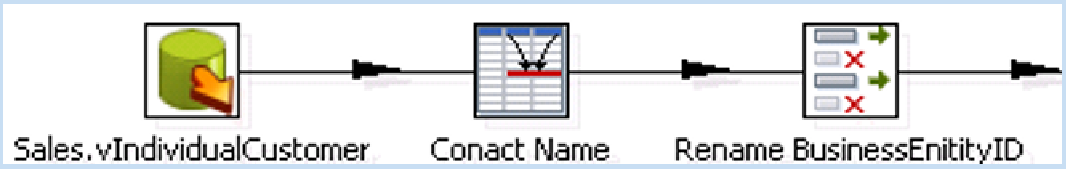
The dimension DimCurrency is generated based on the table Sales.Currency. Only the name and the abbreviation for the respective currency are applied. When this has been done, an identification number (CurrencyID) is generated for the individual tuples, which acts as the primary key. Then, using the step TableDimCurrency, the table is created and populated.



DimCurrency.ktr

DimCustomer

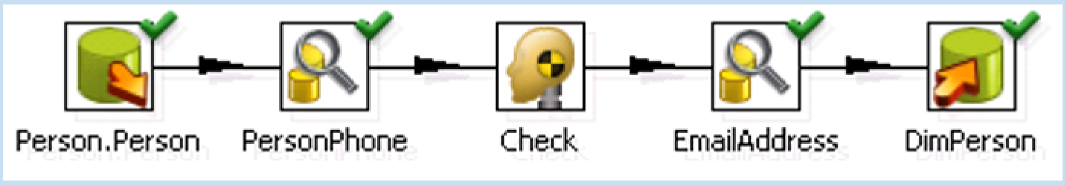
The second transformation, DimCustomer.ktr also does not require a join operation. The dimension is based on the table Sales.vIndividualCustomer. This only requires slight changes with regard to the dimension table DimCustomer, which needs to be created. The first and last name will be combined in a common data field for later analysis. The primary key name is modified to suit the designation within the multi-dimensional data model.



DimCustomer.ktr

DimPerson

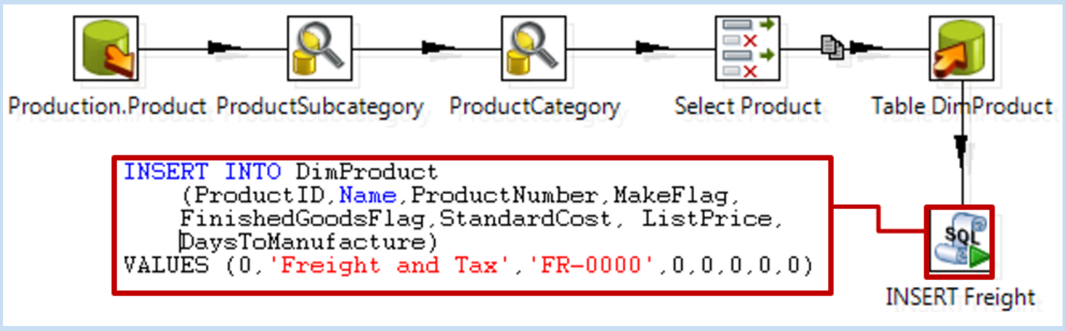
The transformation DimPerson.ktr is based on the table Person.Person. This is expanded to include Person.PersonPhone and Person.EmailAdress using Database Lookup. In both cases, BusinessEntityID is used for comparison as it serves as the primary key for all three tables.



DimPerson.ktr

DimProduct

The dimension DimProduct is generated from three tables of the OLTP database. The transformation starts with the table Production.Product. As a first step, we will include the attributes Name and ProductCategoryID from the table Production.ProductSubcategory. The second of these attributes will serve as the key for the second Lookup operation performed on the table Production.ProductCategory. The next step, Select Product, is used to clean up and rename data fields before populating the table with data. While this is being done, an SQL script is run in parallel to create an additional product data set named Freight. This will enable us to filter on freight costs for individual shipments during our later analyses.

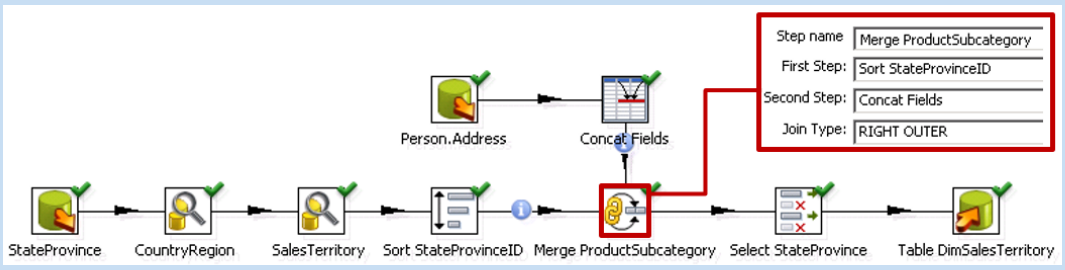


```
INSERT INTO DimProduct
    (ProductID,Name,ProductNumber,MakeFlag,
    FinishedGoodsFlag,StandardCost, ListPrice,
    DaysToManufacture)
VALUES (0,'Freight and Tax','FR-0000',0,0,0,0,0)
```

DimProduct.ktr

DimSalesTerritory

The transformation DimSalesTerritory.ktr essentially corresponds to the previous processes. It is based on the table StateProvince. During the Lookup operation, the attributes Name and Group from the tables CountryRegion and SalesTerritory are added to this table. After this has been successfully completed, the dimension is linked to the table Address using a Merge Join. Prior to this, the address lines within the step Contact Fields are combined, which will prove useful for later analysis. Merge Join is realized as a Right Outer Join, ensuring the address information is only updated for the dimension‘s data sets. In this transformation, Merge Join is, therefore, the most efficient solution.

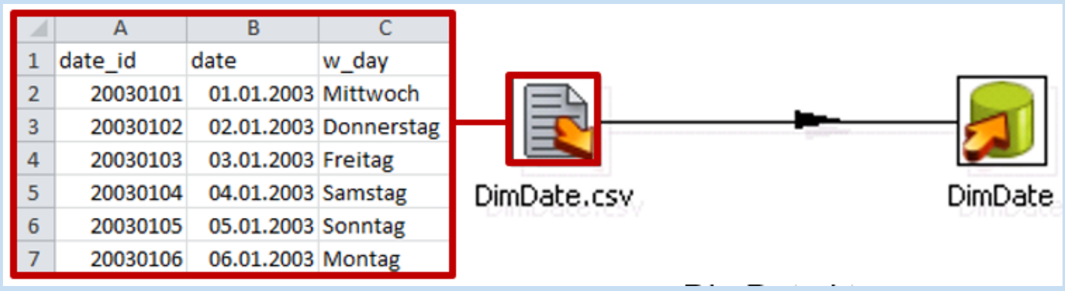


Step name: Merge ProductSubcategory
First Step: Sort StateProvinceID
Second Step: Concat Fields
Join Type: RIGHT OUTER

DimSalesTerritory.ktr

DimDate

The dimension DimDate cannot be extracted from the OLTP database, so it must be created manually. The basis of this transformation is a CSV document created using Excel. The file contains a unique identification number, the specific date, the corresponding day of the week as well as day, month and year – all separated. The dimension also covers the period from 2003 to 2013. The corresponding CSV file can be integrated into Pentaho Data Integration using the step CSV Input File. The data can then be processed and / or loaded into the target database (as depicted in the figure).



	A	B	C
1	date_id	date	w_day
2	20030101	01.01.2003	Mittwoch
3	20030102	02.01.2003	Donnerstag
4	20030103	03.01.2003	Freitag
5	20030104	04.01.2003	Samstag
6	20030105	05.01.2003	Sonntag
7	20030106	06.01.2003	Montag

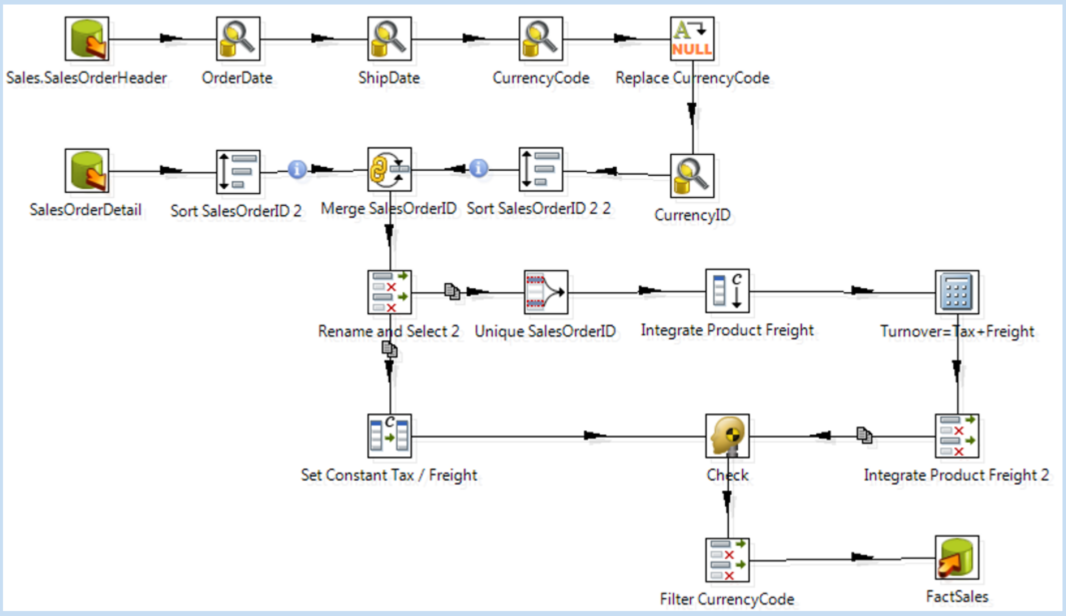
DimDate.ktr

FactSales

This uses the table Sales.SalesOrderHeader. It already contains many important keys and values such as Freight and TaxAmt. The connections are, however, rarely designed to work quickly. This is why OrderDate and ShipDate are replaced by DimDate in the first section, which was generated within the dimension. After this, the attribute ToCurrencyCode from the table Sales.CurrencyRate is integrated using a Lookup. In doing so, CurrencyRateID is used as the key. Since for many orders no currency conversion will be performed (transactions within the dollar zone), the column value is often null. The null values are replaced in the next step by the abbreviation USD for dollars. Then, the attribute is replaced by the better performing CurrencyID from the dimension DimCurrency.

Now, the table SalesOrderDetail can be integrated using a Left Outer Join. This table provides detailed information on the individual orders. This join combines data sets of different aggregation levels, so these will have to be subsequently modified. The data stream is then split for further processing: The aim here is to create an additional data set per order that includes the aggregated values Freight and TaxAmt. This is implemented in the upper path. The lower part, however, processes the data sets for the individual order items. As an alternative, it is also possible to distribute the value of the items Freight and TaxAmt equally among the individual positions of each respective order.

In the lower section, only the two parent values Freight and TaxAmt are set to 0. In parallel to this, the data sets in the upper part are filtered by distinct statement. Then, various attributes such as ProductID or UnitPrice are set to 0. Now, a new value is created from the sum of the taxes and freight costs. This is correspondingly designated as LineTotal. After further terms have been modified, the two data streams are joined again before a final selection of the attributes is carried out. The finished fact table now contains seven keys and six values.



FactSales.ktr

5.3 Data storage & management

Depending on the type of data and the objectives pursued, a number of different database management systems (DBMS) can be integrated at the data management level. Here, a distinction must be made between relational and non-relational systems. Non-relational systems, which are also referred to as NoSQL databases, include MongoDB, HBase and CouchDB. The relational DBMS include traditional, row-oriented products such as MySQL and Postgres as well as analytical, column-oriented systems such as Infobright, InfiniDB and MonetDB.

Within the example architecture we have used Infobright as the database management system for our data warehouse. Infobright is an analytical database based on MySQL. It is, therefore, compatible with a variety of well-known MySQL tools, but also offers significantly better performance when dealing with large amounts of data. At the core of Infobright is the Brighthouse Engine, which has been specially developed for analyzing large data sets. Infobright is available in both a free Community Edition and a paid Enterprise Edition. The Enterprise Edition offers significant advantages in terms of functionality and performance, which is why we will be using it for our prototype data warehouse.

The installation and integration of Infobright is easy due to its high degree of automation. Large parts of the configuration and manual „tweaking“ of the system, including indexing, are omitted. The corresponding settings are carried out by the system automatically, but can also be modified retrospectively. The configuration files my-ib.ini and brighthouse.ini are located in the installation directory.

5.3.1 Performance optimization

Infobright‘s already high performance and compression rate can be enhanced even further. This is important, for example, for the data types Char / Varchar, because they weaken system performance and significantly increase memory requirements. Infobright offers several solutions here, including Lookup Columns and Domain Expert. Both of these are used in our prototype system.

Lookup Columns

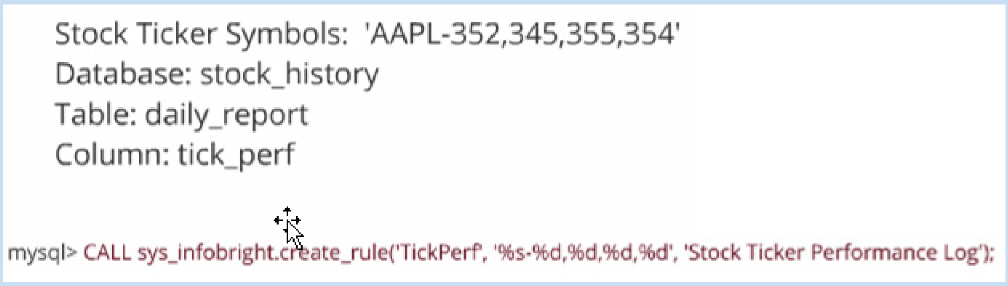
The principle behind Lookup Columns is similar to that of a compressed glossary or dictionary. When using Lookup Columns, all values are transformed into corresponding indices of the type Integer within a column. In parallel, a directory is created that stores the indexes and the corresponding Char / Varchar values. Comparable terms are represented by the same index. This improves performance and reduces the memory requirements of the database.

Lookup Columns are defined when the table is created within the Create statement. Subsequent integration using the Alter statement is not possible. The respective column is completed via the additional „comment ‚lookup‘“. Cardinality plays a crucial role when creating Lookup Columns. For ratios up to 10 to 1, they are definitely worth using. This means that for a column with 100,000 values, a maximum of 10,000 different instances can exist. Thus, for higher cardinalities, any possible gains in performance must be carefully weighed up against a longer process loading time.

DomainExpert

Infobright offers another type of optimization: DomainExpert. This technology enables the user to create an attribute's structure in the form of a rule. By using this technique the way data is processed, stored, and retrieved is sustainably improved. According to the developer, queries will run up to 50% faster and significantly better data compression can be achieved.

DomainExpert can only be used with columns whose values have a particular pattern or a certain structure. For instance, these include, IP addresses (Number.Number.Number.Number) or email addresses (String@String.String):



Stock Ticker Symbols: 'AAPL-352,345,355,354'
Database: stock_history
Table: daily_report
Column: tick_perf

```
mysql> CALL sys_infobright.create_rule('TickPerf', '%s-%d,%d,%d,%d', 'Stock Ticker Performance Log');
```

Example pattern

Only one of the two methods described can be applied per column. In general, Lookup Columns should be preferred. DomainExpert is merely an alternative that can be used if a column has a fixed pattern, but due to a too high cardinality Lookup Columns cannot be used.

Table	Column	Pattern
DimCustomer	EmailAddress	%s@%s.%s
DimDate	Date	%d.%d.%d

DomainExpert rules in our example architecture

5.4 Data analysis

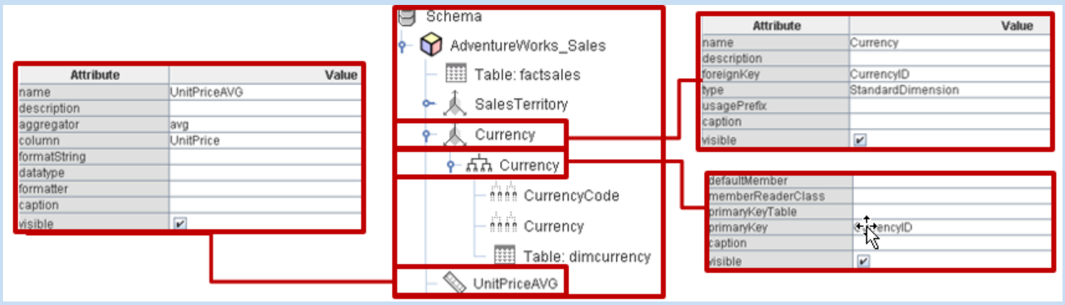
Data analysis within a data warehouse architecture is based on the principle of Online Analytical Processing (OLAP). OLAP makes it possible to carry out dynamic analysis in multi-dimensional spaces, regardless of the physical data storage used. OLAP has a number of variations: relational (ROLAP), multi-dimensional (MOLAP) and hybrid (HOLAP) OLAP.

Within our prototype architecture, we have implemented the data analysis level using the open source server Mondrian. Mondrian is based on the ROLAP or relational approach. Using a data cube, it transforms MDX queries into multiple SQL queries and, conversely, processes relational results multi-dimensionally. In addition, Mondrian uses the information provided by Pentaho Schema Workbench to generate the XML schema. This tool lets you establish a direct connection to the database, which can be used to store attributes in the schema via a drop-down list. The connection is established in the same way as Pentaho Data Integration.

The schema is created according to a fixed structure: The basis here is the cube, which is created by clicking on the cube symbol. Next, the fact table FactSales needs to be defined, after which the individual dimensions can be set up. In addition to the hierarchies and their associated levels, the integration of the respective dimension table must also be performed.

It may be advisable to separate multiple hierarchies of a dimension table in the XML schema into their own dimensions. This allows the hierarchies to be integrated onto the different axes in parallel. We used this principle in our example architecture for the dimensions ShipDate and OrderDate. Special attention should be paid to the specific order of the hierarchy level and / or attributes.

After this has been done, the various key figures can be integrated into the XML schema. Choosing the right aggregator here is crucial. The values Turnover, TaxAmt, OrderQty, Freight are added together using sum. Adding together the values UnitPrice and UnitPriceDiscount would not be meaningful. Instead, we derive their average using avg.



Attribute	Value
name	UnitPriceAVG
description	
aggregator	avg
column	UnitPrice
formatString	
datatype	
formatter	
caption	
visible	✓

Attribute	Value
name	Currency
description	
foreignKey	CurrencyID
type	StandardDimension
usagePrefix	
caption	
visible	✓

defaultMember	
memberReaderClass	
primaryKeyTable	
primaryKey	[illegible]encyID
caption	
visible	✓

Excerpts from the schema

When setting up the schema, special attention must be paid to the specific characteristics of the respective analysis tools. For ad-hoc analyses, Saiku or Pivot4J can be used instead of the Pentaho Analyzer. Pentaho Analyzer relies on the defined hierarchies for labeling dimensions rather than the actual dimensions themselves.

5.5 Data presentation

The data presentation level is the interface between the system and the end user. Various graphical tools can be used to select, analyze, and visualize the data processed by the data warehouse. Included among these are ad-hoc analyses, dashboards, and reports.

For our open source data warehouse, we have chosen to use Pentaho Analyzer and Analyzer Report for the data presentation and visualization layer. Analyzer Report provides a graphical drag and drop interface with which individual fields can easily be added to an analysis. Using various combinations of attributes and values as well as filters, different views of the data can be created. But before this can be done, the schema as well as the corresponding database connection must first be set up within the User Console. This is done via the menu item Manage Data Sources > New Data Source. Once this has been done, it is possible to produce comprehensive analyses of the data stored in the data warehouse.

6. Conclusion and outlook

The architecture illustrated in this document proves that it is possible to construct a high-performance, user-friendly data warehouse system with open source software. Neither during the construction nor the operation of the data warehouse were we able to find any faults, deficiencies or loss of performance for our open source solutions in comparison with products offered by major manufacturers.

We therefore recommend that you take a serious look at open source business analytics software products. There are now a number of excellent open source tools for data analysis and Big Data scenarios that more than compete with the closed source „big boys“. Small and medium-sized businesses can definitely benefit here, because for them, the majority of solutions provided by large software vendors are too powerful and too expensive to use. Large corporate users, by contrast, will benefit from the open architectures, interfaces, and extreme flexibility that an open source data warehouse system can offer.

7. Appendix

Table	Column
• DimDate	• w_day
	• month
• DimCustomer	• Titel
	• City
	• StateProvinceName
	• PostalCode
	• CountryRegionName
• DimPerson	• PersonType
	• Titel
	• FirstName
	• LastName
• DimProduct	• ProductCategoryName
	• ProductSubcategoryName
	• Color
	• ProductLine
	• CLASS
	• Style
• DimSalesTerritory	• City
	• StateProvinceCode
	• StateProvinceName
	• CountryRegionCode
	• LocationName
	• GROUP

Lookup Columns within the example architecture

Leading in Business Open Source solutions and consulting

it-novum is the leading IT consultancy for Business Open Source in the German-speaking market. Founded in 2001 it-novum today is a subsidiary of the publicly-held KAP Beteiligungs-AG.

We operate with 85 employees from our main office in Fulda and branch offices in Düsseldorf, Dortmund, Vienna and Zurich to serve large SME enterprises as well as big companies in the German-speaking markets.

it-novum is a certified SAP Business Partner and longtime accredited partner of a wide range of Open Source products. We mainly focus on the integration of Open Source with Closed Source and the development of combined Open Source solutions and platforms.

Due to the ISO 9001 certification it-novum belongs to one of the few Open Source specialists who can prove the business suitability of their solutions, proven by international quality standards.

More than 15 years of Open Source project experience

- Our portfolio contains a wide range of Open Source solutions within the applications and infrastructure area as well as own product developments which are well-established in the market.
- As an IT consulting company with a profound technical know-how within the Business Open Source area we differentiate ourselves from the big solution providers' standard offerings. Because our solutions are not only scalable and flexible but also integrate seamlessly in your existing IT infrastructure.
- We can assemle multidisciplinary project teams, consisting of engineers, consultants and business data processing specialists. Thus we combine business know-how with technological excellence to build sustainable business processes.
- Our target is to provide you with a high-quality level of consulting during all project phases – from the analysis and conception up to the implementation and support.
- As a decision-making basis prior to the project's start we offer you a Proof-of-Concept. Through a real-case simulation and a developed prototype you can decide on a new software without taking any risks. Moreover, you benefit from:
  - Security and predictability
  - Clear project methodology
  - Sensible calculation

Your contact person for Business Intelligence and Big Data:

Stefan Müller
Director Big Data Analytics
stefan.mueller@it-novum.com
+49 (0) 661 103 942

it-novum GmbH Germany
Headquarters Fulda: Edelzeller Straße 44 · 36043 Fulda
Phone: +49 (0) 661 103 333
Branches in Düsseldorf & Dortmund

it-novum branch Austria
Ausstellungsstraße 50 / Zugang C · 1020 Vienna
Phone: +43 1 205 774 1041

it-novum GmbH Switzerland
Hotelstrasse 1 • 8058 Zurich
Phone: +41 (0) 44 567 62 07