Migrating to Virtual Data Marts using Data Virtualization

Simplifying Business Intelligence Systems

A Technical Whitepaper

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

January 2015

Sponsored by

Table of Contents

1 Introduction

Almost every BI system is made up of many *data marts*. These data marts are commonly developed to improve query performance, to deliver to users the data with the right data structure and the right aggregation level, to minimize network delay for geographically dispersed users, to allow the use of specific database technologies, and to give the users more control over their data.

Unfortunately, data marts are expensive because they require a lot of work to develop, operate, and maintain. They also complicate the architectures of BI systems. For example, changes made to the data warehouse can lead to a multitude of changes throughout all the data marts, and changes made to a report can involve complex modifications to the data marts and the corresponding ETL programs. In addition, they can degrade data quality and complicate data governance.

But most importantly, data marts reduce the flexibility of a BI system. The main reason is that they are *physical data marts*, so they occupy disk storage, they must be loaded with new data periodically, they have to be managed, tuned, and optimized, and so on.

Nowadays, organizations demand *flexible BI systems*. This does not mean that all the data marts should be dropped, because then all their benefits would disappear as well. A better solution is to replace physical data marts by *virtual data marts*. With virtual data marts data can still be delivered with the right structure, in the right form, and at the right data aggregation level. But they are not physical, so they do not occupy disk storage, they do not have to be loaded with new data periodically, they do not have to be managed, tuned, and optimized, and so on. Virtual data marts are much more flexible than physical data marts. Changing them is predominantly a change of specifications.

Virtual data marts can be developed in various ways. This whitepaper describes in detail a step-by-step approach for migrating physical data marts to virtual data marts using *Cisco Information Server* (CIS). The approach is based on an evolutionary migration that does not impact the existing reporting workload.

2 Data Marts

Data Marts – Most BI systems make use of *data marts*; see Figure 1. Some even comprises countless data marts. The data they contain is derived from the data warehouse. ETL programs are used to copy data from the data warehouse to the data marts periodically.

Each data mart is developed for a specific group of users, all with comparable information and reporting needs. A data mart contains a subset of all the data from a data warehouse. Also, whereas a data warehouse contains the lowest level of data, a data mart usually contains a slightly aggregated version of all that data. Most reports run on one of those data marts instead of on the data warehouse.

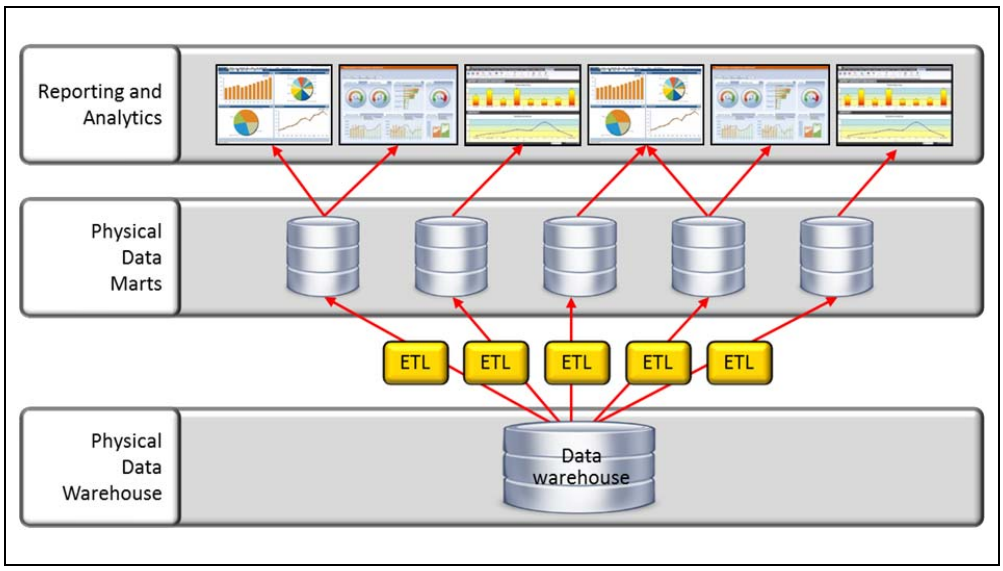

Figure 1 *Traditionally, a BI system consists of many data marts.*

Why Are Data Marts Developed? – Usually, in a BI system the majority of all the reporting and analytics is processed on data marts. Common reasons for developing a data mart are:

- **Query performance:** The most dominant reason for organizations to develop data marts is query performance. Without data marts all the queries must be executed on the data warehouse. This query workload might be too intense for the database server managing the data warehouse, leading to long waiting times for the users. By implementing data marts, most of the queries are offloaded from the data warehouse to these data marts, and by distributing the query workload over multiple data marts, the performance of queries improves.

- **Data structure:** Particular reporting or analytical tools require data to have a certain structure. For example, a tool might demand that the tables are organized as a star schema. If the tables in the data warehouse are normalized, the tool won't be able to access the data. In this case, a data mart is used to present the same data in the right structure. In such a system, ETL scripts are used to transform the normalized data to a star schema arrangement.

- **Geographically dispersed users:** If users are geographically dispersed, querying a central data store might lead to considerable network delay, which slows down the reports. In this situation, it's better to move data physically closer to where the users are located, taking network traffic out of the equation.

- **Storage technology:** Data marts allow the use of different storage technologies for different reports and reporting tools. For example, the performance can be improved for certain reports by storing the data in a multi-dimensional database server in which data is stored in a cube-like form instead of in a SQL database server.

- **Local control:** By having a real copy of their data in a dedicated database, users have more control over that data. For example, it allows them to add private data or data from external sources to enrich their reporting capabilities.

Technologies Used – As indicated, data marts can be developed with different data storage technologies. The most popular form is a SQL database. In this case, a separate set of tables is developed in a SQL database. Technically, this set of tables may be defined inside the database that contains the data warehouse, but quite often a separate database is developed. A second popular form is analytical cubes. Products such as Microsoft Analysis Service and KylinOLAP don't store the data in tables, but in cubes in which the data is organized in dimensions and hierarchies. Data stored in cubes cannot be accessed using SQL, but through a dedicated language, such as MDX. The third option for storing the data of a data mart is by storing it in files, such as Microsoft Excel files, Microsoft Access files, or simple comma-delimited files.

Star Schema – The tables in a data mart developed with SQL technology are usually organized as a *star schema*[1]. In a star schema the tables are classified as *dimension tables* (or *dimensional tables*) and *fact tables*. Fact tables are the central tables in a star schema. A row in a fact table usually represents a *business event*. Examples of potential fact tables are money withdrawals from a bank account, bookings for a flight, and payments at a counter. Each row in a dimension table represents some *business object*, such as a customer, a product, or a department. Dimension tables don't have relationships with each other, but only relationships with fact tables. Star schema owes its name to its graphical representation where the fact table forms the center and the dimension tables are drawn as rays originating from that center, together forming a star.

The primary goal of arranging tables as a star schema is to limit the number of tables that have to be accessed and joined when a query is processed. The often cited advantage of avoiding table joins is improved query performance. Another advantage is that it becomes much easier to write queries and to present the end user with a set of options from which a tool can generate a query. The fact that duplicate data increases the amount of required storage is seen as less important. Also, the fact that duplicate data can lead to inconsistent data is not considered a major disadvantage as well, which makes sense in a data warehouse environment where all the inserts and updates are executed in a very controlled fashion.

A data store can contain many fact tables and thus many star schemas. If fact tables share the same dimension tables, these dimension tables are called *conformed dimension tables*. This is only possible if those fact tables have been designed in such a way that they can use the same dimension tables.

3 The Costs of Data Marts

The Price Tag of Data Marts – Quite often the decision to develop data marts is made lightheartedly. Query performance is commonly used as the all-overriding argument. Unfortunately, data marts are more expensive than most organizations think. Already in 2008, Gartner[2] indicated that the costs of derived data stores, such as data marts, are expensive components of an entire data warehouse architecture. Their recommendation was to "consolidate data marts into an application-neutral data warehouse or smaller data marts to reduce the cost and complexity of the data integration processes feeding the data marts." They predicted that this could save an organization 50% of what they're spending to support siloed data marts.

[1] Wikipedia, *Star schema*; see http://en.wikipedia.org/wiki/Star_schema
[2] Gartner - Friedman, T. et al., *Cost Cutting in Data Management and Integration*, February 2008; see https://www.gartner.com/doc/604907

Data Mart Related Activities – Data marts are expensive because they require many activities to develop, operate, and maintain. For example, developing a new data mart involves at least the following activities:

- Define data structures
- Develop ETL programs to transform and load data
- Create a dedicated database
- Implement the tables
- Optimize and tune the physical storage structure
- Optimize and tune the database server
- Develop backup and recovery processes
- Run the initial load of the tables with data coming from the data warehouse

Next, keeping a data mart operational requires the following activities:

- Periodic refresh of data in the data marts with new data from the data warehouse
- Optimize and tune the database (regularly)
- Optimize and tune the database server
- Optimize and tune ETL programs
- Monitor the database usage
- Run backup processes (and occasionally) recovery processes

Changing an existing data mart also requires a considerable amount of work:

- Temporarily unload data from a data mart
- Change data structure
- Change ETL programs
- Optimize and tune the physical database design
- Optimize and tune the ETL programs
- Reload data in data mart
- Identify and revise the impacted reports

Inflexibility of Data Marts – Data marts do make a BI system less flexible for two main reasons. First, if a report has to be extended with more data, it can mean that the data structures in the data mart must be changed and that the ETL program that loads the data in the data mart must be changed accordingly. In fact all the activities described above may apply. All this requires a considerable amount of work. Second, when a data structure in the data warehouse is changed, data structures in several data marts must be adapted accordingly, plus all the involved ETL programs. In both cases, one change in a report or in the data warehouse demands a long list of activities. This all negatively influences flexibility and thus the time to market of new reports.

Overlapping Data Marts – Another drawback of data marts is that their data contents overlaps. Data marts developed for different user groups may contain the same data. For example, multiple may have a full copy of the customer dimension table. From a storage perspective this is not efficient. But what's more important is that the same data from the data warehouse must be loaded multiple times in several data marts. This slows down the entire load process.

Degrading Data Quality – Duplicating data never improves the data quality. In fact, it increases the risk that data quality degrades. Storing data in a data warehouse and in multiple data marts implies that programs are invoked periodically to keep all these databases up to date. If a new data value is inserted in the data warehouse all the data marts must be kept up to date. Problems can occur when running such programs. For example, they can crash halfway or they may be executed twice unintentionally. Every duplicated version of data is a potential data quality risk. David Loshin[3] worded it as follows: "Multiple copies of the same data lead to more data silos in which the quality of the data continues to degrade, especially when levels of service for consistency and synchronization are not defined or not met." Copying data to multiple data marts can only lead to *entropy* of the BI system and to inconsistency of data.

4 The Alternative: Virtual Data Marts using Data Virtualization

Virtual Data Marts – Traditional data marts as described in Section 2 can be called *physical data marts*, because they occupy disk storage. CIS allows the development of *virtual data marts*. What virtual data marts have in common with their physical counterparts is that both show the data in a form that fits the reporting requirements. For example, in virtual data marts the tables can be organized in the form of a star schemas as well. The key difference between the two is that the contents of the tables in a virtual data mart is not stored. The tables in virtual data marts are defined as virtual tables in CIS.

Benefits of Virtual Data Marts – There are several principal benefits of virtual data marts over physical data marts:

- **Development speed:** Section 3 lists most of the activities for setting up, operating, and changing a physical data mart. It clearly shows how much work is involved. Setting up a virtual data mart requires less activities as is shown in this whitepaper.

- **Agility:** Changing a view defined in a virtual data mart involves altering its definition and nothing more. To make a comparable change in a physical data mart leads to changing the data structures, redefining ETL programs, unloading and reloading data, possibly changing the existing index structures, and re-tuning and re-optimizing the database; see Section 3. In other words, it involves more work and negatively impacts agility. Changing virtual data marts, on the other hand, involves less work and that improves agility.

- **Lower costs:** Developing and changing physical data marts is more expensive than developing and changing virtual data marts. The main reason is, again, less work.

- **Improved data quality:** Because less copies of the data exist when virtual data marts are developed, there is a reduced risk that the data in the copies become inconsistent or incorrect.

- **Improved data governance:** With virtual data marts more data structure definitions and data models are shared. CIS offers lineage and impact analysis making it easy to see all the

[3] D. Loshin, *Effecting Data Quality Improvement through Data Virtualization*, June 2010; see http://purl.manticoretechnology.com/ImgHost/582/12917/2011/resources/white_papers/Effecting_Data_Quality_Improvement_through_Data_Virtualization.pdf

relationships and dependencies between tables and columns in the data warehouse and the vrtual data marts. This simplifies data governance.

- **Simplified data security:** By defining all virtual data marts in CIS, all the data security specifications are centralized. For example, only one set of data acces rules have to be defined, and not for each physical data mart a separate set.

A potential advantage of a physical data mart is that its performance can be better than that of its virtual counterpart. However, several techniques exist to improve the performance of a virtual data mart, such as enabling caching; see Section 5.2. In this case, a comparable performance can be obtained.

5 Developing Virtual Data Marts

This section describes step by step how to migrate an existing physical data mart to a virtual data mart without impacting the operational reports:

1. Recreating physical data marts as virtual data marts
2. Improving query performance on virtual data marts
3. Identifying common specifications among virtual data marts
4. Redirecting reports to access virtual data marts
5. Extracting definitions from the semantic layer of the reporting tools and placing them in view definitions
6. Defining security rules
7. Adding external data to virtual data marts

We assume that all the physical data marts are developed with SQL database servers, such as Oracle, Microsoft SQL Server, or MySQL.

Throughout this section the same sample data warehouse is used. Figure 2 shows the database structure of this data warehouse. This example is based on the example used in the book *Data Virtualization for Business Intelligence Systems*[4].

Step 1: Recreating Physical Data Marts as Virtual Data Marts

Importing Data Warehouse Tables – To be able to develop a virtual data mart, the tables in the existing data warehouse from which the physical data mart is loaded, must be imported in CIS. Identifying these tables may be easy if an ETL tool is used to extract data from the data warehouse. Most of these products support lineage analysis with which the relationships between data warehouse tables and data mart tables can be clearly presented. When internally-developed code is used, this step may involve digging deep in the code.

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

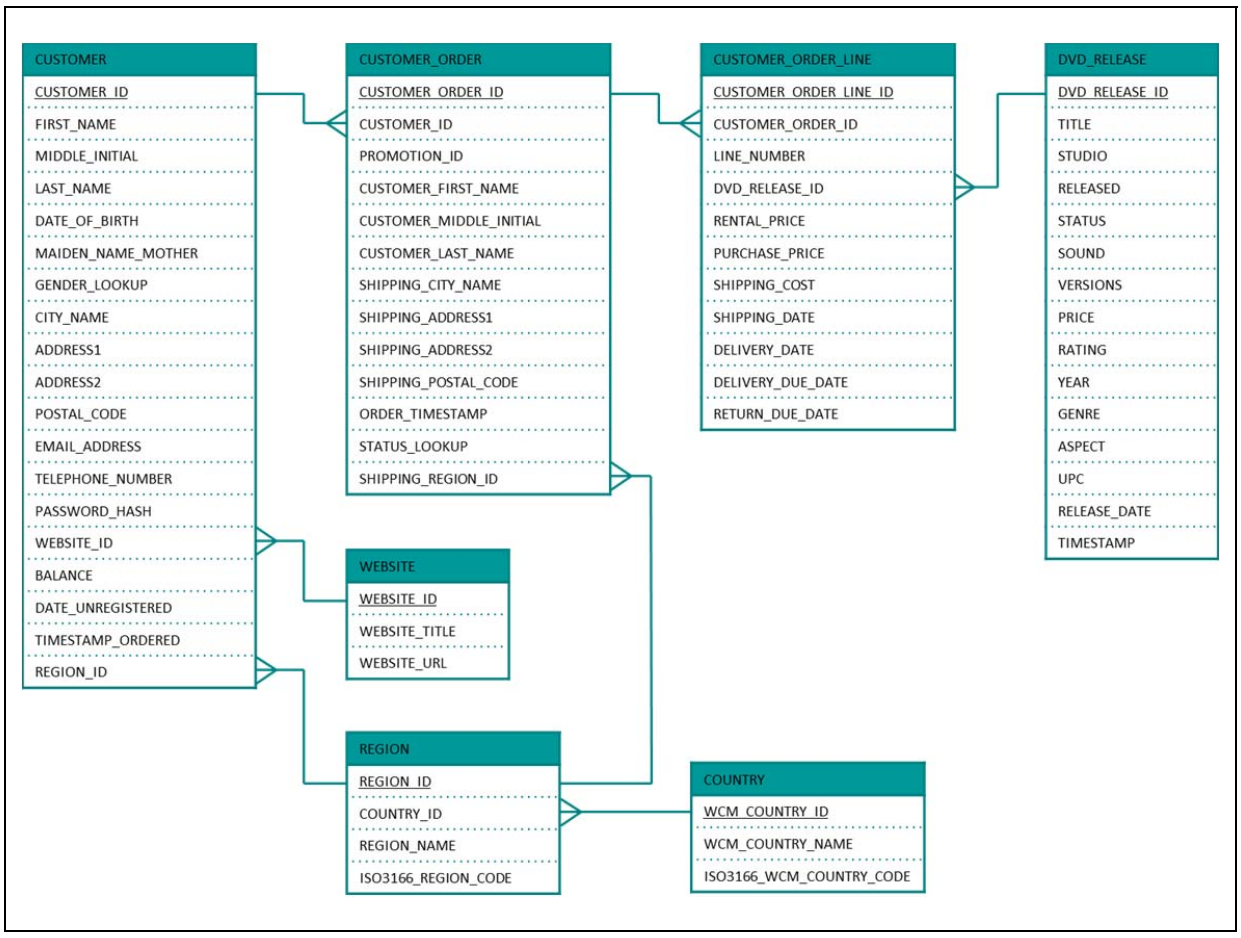

Figure 2 *The data model of a data warehouse.*

Defining Virtual Data Marts – Next, views must be defined with the same structure as the tables in the physical data mart. The logic to define the views can be derived from the ETL program that loads data into the physical data mart. To guarantee a 1:1 conversion, the view logic must be equivalent to the ETL logic so that the reports return the same results when they have been migrated to CIS.

Unfortunately, no tools exist to perform this migration automatically. This entire step is executed manually. Most ETL tools offer a high-level, flow-like language for specifying how to transform data structures and data values. Familiar operations are sort, filter, join, and aggregate that all have a corresponding operation in SQL and are therefore easy to migrate. Besides looking at the ETL programs, it's worthwhile to study the source-to-consumer mappings from which the ETL was developed. These mappings may be more conceptual and less product-dependent.

The result of Step 2 is a solution in CIS consisting of two levels of views; see Figure 3.

Migrating a Star Schema – As indicated, the tables of many data marts are organized as a star schema. For example, the data in the data warehouse depicted in Figure 2 may be transformed to the star schema presented in Figure 4.

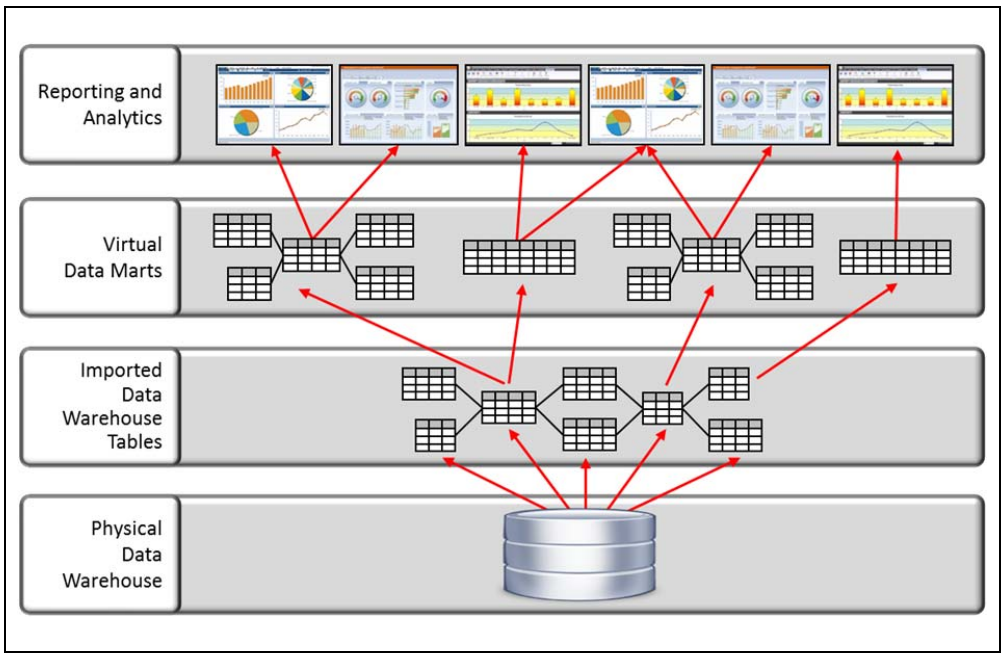

Figure 3 *The approach described in this section results in two levels of views in CIS; the two middle layers in this diagram.*

In most cases such a transformation is relatively straightforward. For example, the view definition for the `CUSTOMER` dimension is a join of four tables (note that VDM stands for Virtual Data Mart):

```
DEFINE   VIEW VDM_CUSTOMER AS
SELECT   C.CUSTOMER_ID,
         C.FIRST_NAME,
         C.MIDDLE_INITIAL,
         C.LAST_NAME,
         C.DATE_OF_BIRTH,
         C.MAIDEN_NAME_MOTHER,
         C.GENDER_LOOKUP,
         C.CITY_NAME,
         C.ADDRESS1,
         C.ADDRESS2,
         C.POSTAL_CODE,
         C.EMAIL_ADDRESS,
         C.TELEPHONE_NUMBER,
         C.PASSWORD_HASH,
         C.WEBSITE_ID,
         W.WEBSITE_TITLE,
         W.WEBSITE_URI,
         C.BALANCE,
         C.DATE_REGISTERED,
         C.DATE_UNREGISTERED,
         C.TIMESTAMP_CHANGED,
         C.REGION_ID,
         R.REGION_NAME,
         R.ISO3166_REGION_CODE,
         R.COUNTRY_ID,
         N.WCM_COUNTRY_NAME,
         N.ISO3166_WCM_COUNTRY_CODE
FROM     CUSTOMER AS C
            LEFT OUTER JOIN WEBSITE AS W ON C.WEBSITE_ID = W.WEBSITE_ID
            LEFT OUTER JOIN REGION AS R ON C.REGION_ID = R.REGION_ID
            LEFT OUTER JOIN COUNTRY AS N ON R.COUNTRY_ID = N.WCM_COUNTRY_ID
```

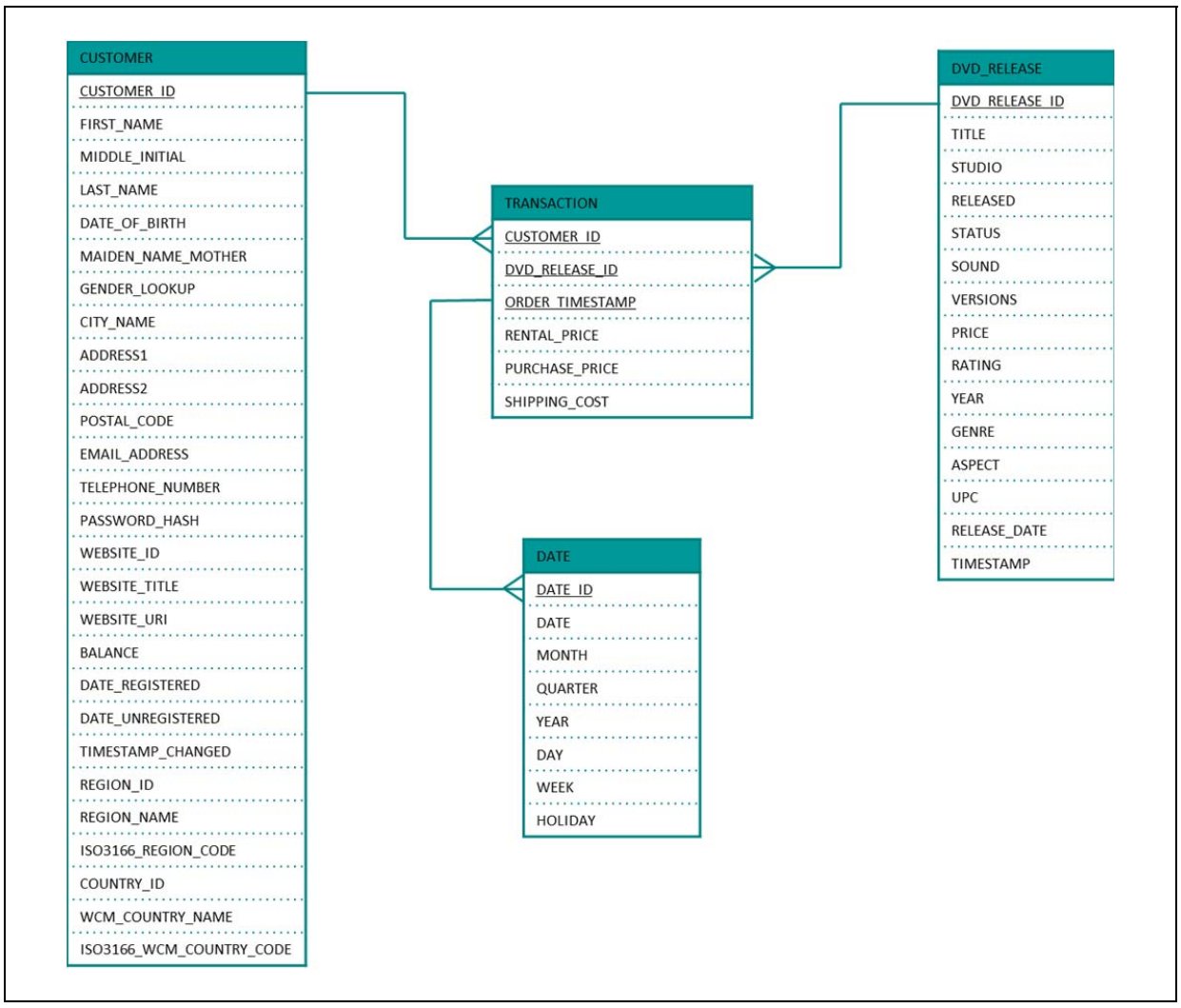

Figure 4 *The data model of a data mart.*

The definition of the fact table called `TRANSACTION` is a join of two tables:

```
DEFINE   VIEW VDM_TRANSACTION AS
SELECT   CUO.CUSTOMER_ID,
         COL.DVD_RELEASE_ID,
         CUO.ORDER_TIMESTAMP,
         COL.RENTAL_PRICE,
         COL.PURCHASE_PRICE,
         COL.SHIPPING_COST
FROM     CUSTOMER_ORDER_LINE AS COL
            LEFT OUTER JOIN CUSTOMER_ORDER AS CUO
            ON COL.CUSTOMER_ORDER_ID = CUO.CUSTOMER_ORDER_ID
```

Avoid Duplicate Views – As indicated, multiple data marts may contain identical tables. For example, they may all contain a copy of the `CUSTOMER` table. In that case, do not create two views, but just one. If two tables are almost identical, defining one view is probably sufficient. For example, if the only difference is that one has a few columns less than the other, define one view that includes all the columns and hide the extra columns for one user group.

Missing Tables – A data mart may contain a table that does not contain data from the data warehouse. For example, there may be a table called `DATE`; see Figure 4. Such tables have been developed to speed up date/time-related queries. If such tables exist, implement them in the data warehouse as physical tables and define views on top in CIS.

Complex Code – Some parts of ETL programs may be hard to migrate to CIS. Here are some potential problem areas:

External code: If an ETL program invokes external code, try to replace this code by CIS procedural code. If this is not possible and if the external code supports an API supported by CIS, try to invoke the code from within CIS.

Complex cleansing functions: It's quite unusual that ETL programs invoke cleansing functions when data is copied from the data warehouse to data marts. Usually, this is done before the data is copied into the data warehouse. Still, it may be possible. If these functions are supported by the ETL tool, it may be difficult to replace them by CIS functions, because the logic may just be too complex. If the functions can be accessed using SOAP or REST interfaces, it may be possible to invoke them in the same way using CIS. As a last resort, retain a limited set of ETL programs and physical copies and make CIS access them.

Procedural logic: The code to transform the structure of the data warehouse tables to the data mart tables may be procedural. This means that the code has not been written in a declarative language such as SQL. The reason may be that the transformations are very complex. Imagine that a column in a data warehouse table contains values that represent EDIFACT messages. An example of such a value is:

```
UNB+UNOA:1+005435656:1+006415160:1+060515:1434+00000000000778'XXXUNH+
00000000000117+INVOIC:D:97B:UN'XXXBGM+380+342459+9'XXXDTM+
3:20060515:102'XXXRFF+ON:521052'XXXNAD+BY+792820524::16++
CUMMINSMIDRANGEENGINEPLANT'XXXNAD+SE+005435656::16++
GENERALWIDGETCOMPANY'XXXCUX+1:USD'XXXLIN+1++157870:IN'XXXIMD+
F++:::WIDGET'XXXQTY+47:1020:EA'XXXALI+US'XXXMOA+203:1202.58'XXXPRI+
INV:1.179'XXXLIN+2++157871:IN'XXXIMD+F++:::DIFFERENTWIDGET'XXXQTY+
47:20:EA'XXXALI+JP'XXXMOA+203:410'XXXPRI+INV:20.5'XXXUNS+S'XXXMOA+
39:2137.58'XXXALC+C+ABG'XXXMOA+8:525'XXXUNT+23+00000000000117'XXXUNZ+
1+00000000000778'
```

Turning this value in a clean set of view columns is hard to do with SQL. Using a piece of procedural code may be more efficient. In such a situation, develop a view using a stored procedure with scripting code. It's also worth studying whether the transformation of this data can be moved to the ETL program that loads this data in the data warehouse.

Proprietary database server functions: ETL programs may have used proprietary database server functions. For example, MySQL supports the proprietary functions `DATABASE` (returns the name of the accessed database) and `VERSION` (returns the identification of the version number of MySQL, something like `5.0.7-beta-nt`). If proprietary functions are used, try to implement a CIS function that returns a similar value.

Tables with Incorrect Data – For an ETL program it's not uncommon to store *incorrect records* in a separate file or table when data is copied from a source to a target. Incorrect records contain data that does not comply to particular rules. When incorrect records are written to a separate file, it's almost as if the ETL program returns multiple results: the tables containing the correctly copied records and tables containing

the incorrect records. Afterwards, the file with the defective records can be sent to the business users so that they can correct the data.

A view in CIS can't have two outputs. A view can be defined to "stop" all the incorrect data to be presented, but it can't have a second output. If BI specialists need to see whether the data warehouse tables do contain incorrect data, a complementary view can be defined that shows the incorrect data.

Example: Imagine that the following two integrity rules apply to the VDM_DVD_RELEASE view:

- Date values in the RELEASE_DATE column must all be greater than December 31, 1989, because the company didn't exist before that date.
- The column STATUS must have one of the following six values: Cancelled, Discontinued, Out, Pending, Postponed, or Recalled.

Here is the definition of the data mart view containing the two intregity rules and showing correct data:

```
DEFINE   VIEW VDM_DVD_RELEASE AS
SELECT   *
FROM     DVD_RELEASE
WHERE    RELEASE_DATE > '1989-12-31'
AND      STATUS IN ('Cancelled','Discontinued','Out','Pending','Postponed','Recalled')
```

Explanation: All the rows with a timestamp that is too old and those whose status is not in the list of allowed values, are excluded from the result. So, the users don't see those rows.

For BI specialists a second view can be defined that shows the rows with the incorrect rows. There are two ways to do this. With the first solution the view definition contains the two integrity rules that apply to the data in the DVD_RELEASE table:

```
DEFINE   VIEW VDM_DVD_RELEASE_DEFECTIVE AS
SELECT   *
FROM     DVD_RELEASE
WHERE    NOT (RELEASE_DATE > '1989-12-31' AND
         STATUS IN ('Cancelled','Discontinued','Out','Pending','Postponed','Recalled'))
```

With the second solution the view definition does not require the rules to be specified twice:

```
DEFINE   VIEW VDM_DVD_RELEASE_DEFECTIVE AS
SELECT   *
FROM     DVD_RELEASE
EXCEPT
SELECT   *
FROM     VDM_DVD_RELEASE
```

If VDM_DVD_RELEASE is redefined, automatically, the definition of VDM_DVD_RELEASE_DEFECTIVE is changed. The first solution probably has a better performance, but requires the rules to be specified in both views complicating their maintenance.

Setting the Refresh Rate – Be careful with the refresh rate of virtual data marts. The frequency with which data warehouses are refreshed may differ from the one used for the physical data marts. For example, the refresh rate a data warehouse may be every 24 hours, whilst a data mart is refreshed every 48 hours. If

the data warehouse is refreshed more frequently than a data mart, the view definition must be extended to simulate this lower refresh rate. This is necessary to guarantee identical report results.

The extension consists of two parts. First, a table must be created in the data warehouse that keeps track for each view the date and time of the (hypothetical) last refresh. For example, this table may look like this:

```
VIEW                          LAST_REFRESH
----------------------------  -------------------
VDM_CUSTOMER_ORDER_IN_01_AREA  2015-01-06 23:59:59
VDM_CUSTOMER_ORDER_LINES       2015-01-07 23:59:59
:                             :
```

Every time when the data warehouse is refreshed, the `LAST_REFRESH` values must be updated. Note that a more sophisticated solution can be developed.

Next, a filter must be added to the view definition:

```
DEFINE   VIEW VDM_CUSTOMER_ORDER_IN_01_AREA AS
SELECT   *
FROM     DW_CUSTOMER_ORDER
WHERE    SHIPPING_POSTAL_CODE LIKE '01%'
AND      LOAD_TIMESTAMP <
        (SELECT LAST_REFRESH FROM REFRESH_DATES WHERE VIEW = 'VDM_CUSTOMER_ORDER_IN_01_AREA')
```

We assume a column called `LOAD_TIMESTAMP` exists in the `DW_CUSTOMER_ORDER` table and contains the date and time indicating when this record was loaded in the data warehouse. The extra filter guarantees that the view will only show those records of the `DW_CUSTOMER_ORDER` table that were loaded before the last refresh cycle of the view.

Defining Primary and Foreign Keys – Define the same *primary* and *foreign keys* on the data mart views as have been defined on the tables in the physical data mart. Although these keys have no meaning for CIS, it's recommended to define them, because many reporting and analytical tools need to know the primary and foreign keys of the tables. When these keys are defined, they can be exposed as metadata and can be retrieved by the tools. Due to the keys, the tools understand the structure of the tables better and they can generate more efficient queries.

Step 2: Improving Query Performance on Virtual Data Marts

The performance of queries on the data mart views depends on many aspects, including the hardware being deployed, the database server in use, the amount of data, whether caching is switched on, the size of the virtual contents of the views, and the query workload. This section contains some recommendations for optimizing the performance of a virtual data mart when performance problems are encountered. Note that these are very generic recommendations.

Updating Statistical Information Of Data Warehouse Tables – A classic reason why performance may be poor is that the *statistical information* on the tables and indexes in the data warehouse are out of date; they don't reflect the correct situation anymore. Because the query optimizer of a SQL database server bases its processing strategy on this statistical information, it's crucial that it's always up to date. How statistical

information can be updated depends on the product. Therefore, we refer to the manuals.

Caching of Data Mart Views – For the data mart views that exhibit a poor query performance, define a cache. For each cached view, a refresh frequency must be defined. Use the frequency that is used currently to refresh the physical data mart to guarantee identical reporting results.

By default, CIS implements cached views with files. Access to those files can be fast, but it's not the fastest data storage technology available, especially if the files become large. Therefore, it's recommended that the caches are stored in a SQL database server.

CIS supports caches to be stored in several SQL database servers, such as EMC Greenplum, HP Vertica, IBM DB2 LUW, IBM PureData for Analytics (Netezza), Microsoft SQL Server, MySQL, Oracle, PostgreSQL, SAP Sybase ASE, Sybase IQ, and Teradata. Most of these are much more scalable and high-performing than files and definitely speed up the queries on cached views.

Indexes on Cached Views – The benefit of using a SQL database server is that all its performance improving features can be deployed to speed up access to the caches, such as memory buffer optimization, table partitioning, and tablespace organization. One of those features is that extra indexes can be defined on the caches to speed up query performance.

Guidelines on defining indexes:

- Define indexes on the foreign keys in the data mart tables to speed up joins.
- Define indexes on columns that are aggregated often.
- Define indexes on columns on which many filters are applied.

Note that adding extra indexes can slow down the refreshing of the caches somewhat.

Developing the indexes on the caches can't be done from within CIS. This must be done using the SQL database server itself. When an extra index is defined, be sure that the right parameters are set; see the manuals of the SQL database server.

Note: The structure of a table used for caching a view is a little different from the view itself. CIS adds one column called `CACHEKEY` for internal processing. It contains an identifier of the refresh run. A separate table indicates what that identifier means. Never change anything with respect to this column. Nor should the indexes that are defined by CIS itself be dropped.

Updating Statistical Information Of Caches – As it's important that the statistical information for data warehouse tables is up to date, it's also important that statistical information on the cached views is up to date. Be sure that after every refresh of a cache an update of the statistical information is considered. Out-of-date statistical information can lead to poor query performance.

Optimizing the SQL Database Server – Each SQL database server offers a wide range of features to improve performance. Tables can be partitioned, buffers can be enlarged and tuned, table space parameters can be optimized, data can be moved to faster storage technology, and so on. Check with database administrators to see if they can tune and optimize access to the cached views.

Migrating to Another Database Server – If possible, another database server can also be selected to run the data warehouse itself. The performance may be structurally poor, because the database server used for the data warehouse is not up to the job. In this case, a migration of the data warehouse to one of the analytical SQL database servers, such as EMC Greenplum, HP Vertica, IBM PureData for Analytics (Netezza), SAP Sybase IQ, and Teradata, can be considered.

Step 3: Identifying Common Specifications Among Virtual Data Marts

After processing the first two steps, the result consists of two levels of views; see also Figure 3. Many of the data marts views will undoubtedly contain identical or similar definitions. It is recommended to extract the *common specifications* and *definitions* from the views and to implement them in a centralized fashion by defining them on a middle layer of views; see Figure 5.

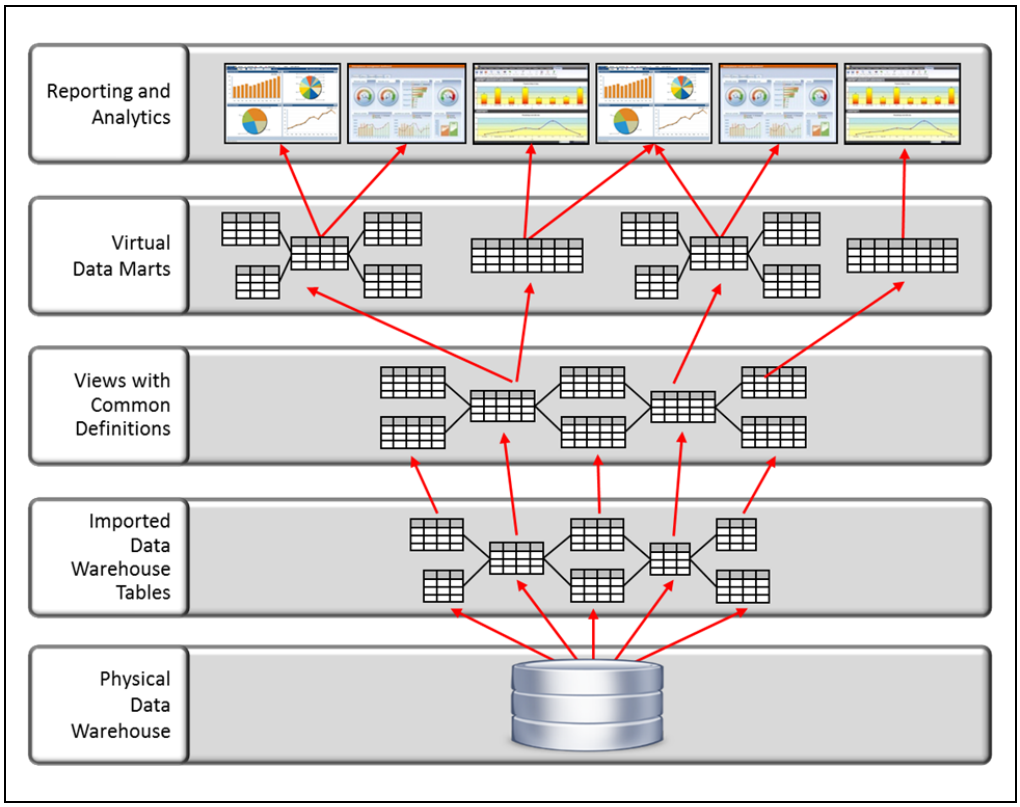

Figure 5 *The middle layer of views contains common specifications.*

By centralizing the common specifications in a set of views, they can be shared and that simplifies maintenance. In addition, new views on the data mart layer won't have to be developed from scratch, because existing specifications in the common layer can be reused. In fact, this step can result in identifying *reference data* and *master data* common to all the virtual data marts. Identifying common specifications is an investment in the future. It leads to the development of a canonical data model and simplifies data governance.

To illustrate this, take the following two definitions of data mart views:

```
DEFINE   VIEW VDM_SOME_CUSTOMERS AS
SELECT   CUSTOMER_ID, LAST_NAME, YEAR(DATE_OF_BIRTH), CITY_NAME
FROM     CUSTOMER
WHERE    YEAR(DATE_OF_BIRTH) > 1980
AND      POSTAL_CODE LIKE '4%'

DEFINE   VIEW VDM_CUSTOMERS_PER_CITY AS
SELECT   CITY_NAME, COUNT(*)
FROM     CUSTOMER
WHERE    YEAR(DATE_OF_BIRTH) > 1980
AND      POSTAL_CODE LIKE '4%'
GROUP BY CITY_NAME
```

Both views contain identical specifications, such as `YEAR(DATE_OF_BIRTH) > 1980` and `POSTAL_CODE LIKE '4%'`. The specifications can be extracted from these views and can be defined in a new view (CV stands for Common Views):

```
DEFINE   VIEW CV_SOME_CUSTOMERS AS
SELECT   CUSTOMER_ID, LAST_NAME, DATE_OF_BIRTH, CITY_NAME
FROM     CUSTOMER
WHERE    YEAR(DATE_OF_BIRTH) > 1980
AND      POSTAL_CODE LIKE '4%'
```

Next, the two views are changed accordingly:

```
DEFINE   VDM_SOME_CUSTOMERS AS
SELECT   CUSTOMER_ID, LAST_NAME, YEAR(DATE_OF_BIRTH), CITY_NAME
FROM     CV_SOME_CUSTOMERS

DEFINE   VDM_CUSTOMERS_PER_CITY AS
SELECT   CITY_NAME, COUNT(*)
FROM     CV_SOME_CUSTOMERS
GROUP BY CITY_NAME
```

In this example, if the filter `POSTAL_CODE LIKE '4%'` has to be changed to `POSTAL_CODE LIKE '4%' OR POSTAL_CODE LIKE '5%'`, it requires just one change in the `CV_SOME_CUSTOMERS` view, and not two. Especially in large environments with hundreds of views, this improves development time and leads to more consistent report results.

To make the entire solution more manageable and easier to maintain, try to locate these identical and similar definitions. It requires in-depth knowledge of all the view definitions. *CIS Discovery* can be used to investigate whether relationships exist between views. This tool locates key entities and reveals relationships between tables.

Step 4: Redirecting Reports to Access Virtual Data Marts

In this step the reports are redirected from the physical data marts to the views defined in CIS. Instead of using the ODBC/JDBC driver of the SQL database running the data mart, they must access data through one of CIS' ODBC/JDBC drivers. This mainly involves work on the side of the reporting tool. For example, changes may have to be made to a semantic layer or in a metadata directory.

Except for some minor issues, for the majority of the reports this migration will proceed effortlessly. For example, it could be that the reports execute SQL statements that use proprietary features not supported by CIS. In this case, try to rewrite these SQL statements and replace the proprietary features by more standard ones. If specific scalar functions are missing, develop them with CIS. These functions are executed by CIS itself and not by the underlying SQL database server.

Note: Execute this step report by report. First, apply this step to one report before moving on to the next. This way, lessons learned from migrating one report can be used when migrating the next one.

Step 5: Extracting Definitions from the Reporting Tools

Many tools for reporting and analytics allow users to define data-related specifications inside their tools. For example, a user can define alternative names for columns, introduce derived columns, add filters on rows, define aggregated columns, and so on. Defining such specifications with the reporting tools has three drawbacks:

- Developing these specifications costs the business users valuable time.
- These specifications may not be formally tested properly.
- The specifications are seldom shared across users. This can lead to inconsistent specifications and different report results.

Extract these definitions from the tools and implement them in the data mart views. Then, they can be formally tested and shared. For example, if a user has implemented a formula to derive the age of an employee based on his birth date, it's probably easier to add this formula as an extra column to the view definition. Other users can then use that same age field as well.

Step 6: Defining Security Rules

Often, users have been granted query access to all the data stored in the data mart they access. A similar set of authorization privileges must be defined in CIS. The following steps are recommended:

- Define a user group for each original physical data mart. The name of this user group can be comparable to the original name of the physical data mart.
- For each user group assign `SELECT` privilege on all the data mart views that represent tables that were once part of the corresponding physical data mart. For example, if the physical data mart contained a table called `ORDERS`, `SELECT` privilege is assigned to the user group on the view called `ORDERS`.

- Assign each individual user to the user groups that correspond with the physical data marts they were allowed to access. Some users may be member of multiple user groups and they inherit and accumulate rights from all the groups to which they belong.

Step 7: Adding External Data to Virtual Data Marts

This step deals with all the data integration specifications that users have created themselves. Many tools allow users to blend data from the physical data mart with data stored in some private files, such as spreadsheets and MS Access files. These integration specifications are not commonly shared amongst users, but are private developments. As in Step 5, try to define these integrations inside CIS so that they can be shared among users.

Do not do this for the incidental integration projects, but for the more structural ones. For example, define integrations in CIS when they are used by several users, have been used for some time, and have been used continuously.

This step is hard to formalize, but it's worthwhile doing so that wheels are not invented over and over again.

6 Getting Started

When all the steps are executed, it's recommended to execute them in the order specified in this whitepaper. However, it's not required to execute all the steps right away. Steps can be skipped first and executed later. Figure 6 shows which steps are optional and which ones are mandatory; Steps 1, 2, and 4 are mandatory and the others optional. For example, in a project the decision can be made to first focus on the three mandatory steps plus Step 6 to secure the data. After these steps, the new system is made available to the users, and only after a few weeks or months after the system has proven itself with respect to performance, stability, and the correctness of report results, are the remaining steps executed.

It's not recommended to migrate all the physical data marts on one go. Such an approach would be too risky. Make it an iterative process. Finish the migration of one physical data mart before the next migration project is started. This gradual and evolutionary makes it easier to reuse specifications.

Final remark: After all the physical data marts have been migrated, not all the work is done. Steps 3, 5, and 7 are never finished. Business analysts can always try to improve the entire set of view definitions. Aim for more sharing of specifications. Over time specifications will migrate gradually from the reports to the top layer views and onwards to the lower layer views.

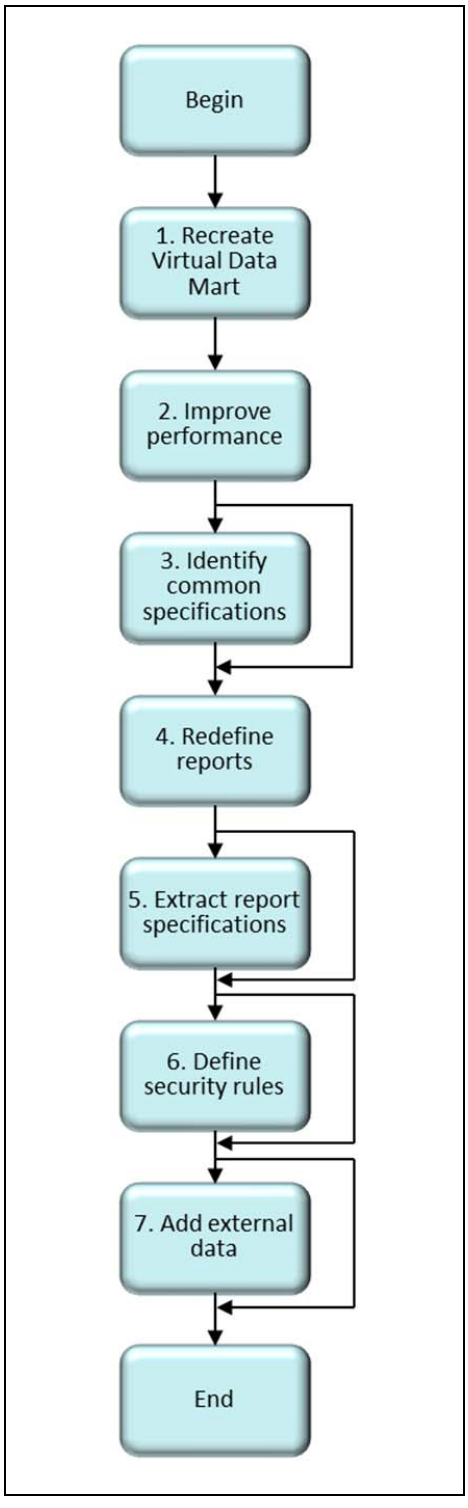

Figure 6 *All the steps to migrate a physical data mart to CIS.*

About the Author Rick F. van der Lans

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

Rick is chairman of the annual European Enterprise Data and Business Intelligence Conference (organized annually in London). He writes for Techtarget.com[5], B-eye-Network.com[6] and other websites. He introduced the business intelligence architecture called the *Data Delivery Platform* in 2009 in a number of articles[7] all published at BeyeNetwork.com. The Data Delivery Platform is an architecture based on data virtualization.

He has written several books on SQL. Published in 1987, his popular *Introduction to SQL*[8] was the first English book on the market devoted entirely to SQL. After more than twenty five years, this book is still being sold, and has been translated in several languages, including Chinese, German, and Italian. His latest book[9] *Data Virtualization for Business Intelligence Systems* was published in 2012.

For more information please visit www.r20.nl, or email to rick@r20.nl. You can also get in touch with him via LinkedIn and via Twitter @Rick_vanderlans.

About Cisco Systems, Inc.

Cisco (NASDAQ: CSCO) is the worldwide leader in IT that helps companies seize the opportunities of tomorrow by proving that amazing things can happen when you connect the previously unconnected. Cisco Information Server is agile data virtualization software that makes it easy for companies to access business data across the network as if it were in a single place.

For more information, please visit www.cisco.com/go/datavirtualization.

[5] See http://www.techtarget.com/contributor/Rick-Van-Der-Lans
[6] See http://www.b-eye-network.com/channels/5087/articles/
[7] See http://www.b-eye-network.com/channels/5087/view/12495
[8] R.F. van der Lans, *Introduction to SQL; Mastering the Relational Database Language*, fourth edition, Addison-Wesley, 2007.
[9] R.F. van der Lans, *Data Virtualization for Business Intelligence Systems*, Morgan Kaufmann Publishers, 2012.