Institute for Parallel and Distributed Systems / AS

Modeling Data Lakes with Data Vault

Practical Experiences, Assessment, and Lessons Learned

Corinna Giebler, Christoph Gröger, Eva Hoos, Holger Schwarz, and Bernhard Mitschang

In: Proceedings of the 38th Conference on Conceptual Modeling (ER 2019)

BIBTEX:
@inproceedings{Giebler2019_DataVault_ER,
author = {Giebler, Corinna and Gr{\"{o}}ger, Christoph and Hoos, Eva and Schwarz, Holger and Mitschang, Bernhard},
booktitle = { Proceedings of the 38th Conference on Conceptual Modeling (ER 2019)},
title = {{ Modeling Data Lakes with Data Vault: Practical Experiences, Assessment, and Lessons Learned}},
year = {2019},
doi={10.1007/978-3-030-33223-5_7}
}

Modeling Data Lakes with Data Vault: Practical Experiences, Assessment, and Lessons Learned

Corinna Giebler[1][0000-0002-5726-0685], Christoph Gröger[2][0000-0001-6615-4772], Eva Hoos[2], Holger Schwarz[1], and Bernhard Mitschang[1]

[1] University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany
{Firstname.Lastname}@ipvs.uni-stuttgart.de
[2] Robert Bosch GmbH, Borsigstraße 4, 70469 Stuttgart, Germany
{Firstname.Lastname}@de.bosch.com

Abstract. Data lakes have become popular to enable organization-wide analytics on heterogeneous data from multiple sources. Data lakes store data in their raw format and are often characterized as schema-free. Nevertheless, it turned out that data still need to be modeled, as neglecting data modeling may lead to issues concerning e.g., quality and integration. In current research literature and industry practice, Data Vault is a popular modeling technique for structured data in data lakes. It promises a flexible, extensible data model that preserves data in their raw format. However, hardly any research or assessment exist on the practical usage of Data Vault for modeling data lakes. In this paper, we assess the Data Vault model's suitability for the data lake context, present lessons learned, and investigate success factors for the use of Data Vault. Our discussion is based on the practical usage of Data Vault in a large, global manufacturer's data lake and the insights gained in real-world analytics projects.

Keywords: Data Lakes, Data Vault, Data Modeling, Industry Experience, Assessment, Lessons Learned.

1 Introduction

The advance of digitalization leads to large amounts of heterogeneous data. Businesses that apply data analytics on these data can gain a large competitive advantage [1]. Data lakes [2] are highly popular, since they enable the integration and explorative analysis of heterogeneous data. Typically, a *schema-on-read* approach is used to manage data in data lakes [2, 3] to allow flexible usage beyond predefined use cases—so called use-case-independence. Even though data of any format may be stored in the data lake, the majority of data lakes in industry practice nowadays mostly contain structured data [4].

However, when managing data with the schema-on-read approach, data modeling must not be neglected [5, 6]. It turned out that a lack of meaningful structure for data may lead to quality issues, integration issues, performance issues and deviations from enterprise goals [6]. Standardizing data modeling in data lakes has two advantages for

organizations: Technical and organizational processes (e.g., for ETL and project management) can be reused, and data from different contexts can easily be combined.

One candidate for modeling data in data lakes is *Data Vault* [7, 8]. It is used to model data lakes in both research and industry practice. Data Vault is a combination of dimensional modeling and third normal form [7] and supports agile project management and use-case-independent modeling [8, 9]. Because it is a simple and flexible modeling technique, Data Vault qualifies for data modeling in data lakes [5].

Currently, there is little conceptual work on Data Vault available in both industry and research. Aside from the reference books of its inventor [7, 8], there are some rudimentary comparisons between Data Vault and other modeling techniques [9, 10]. Research also deals with the creation of a conceptual Data Vault model [11], the automated physical design of Data Vault [12], or the direct transformation from JSON to a Data Vault schema [13]. However, there are neither insights on practical experiences nor detailed assessments for Data Vault, especially not in the context of data lakes.

In this paper, we close this gap by providing guidance on the usage of Data Vault in data lakes. Our contributions include the following:

- We investigate exemplary real-world analytics projects from three different core business domains at a large, globally active manufacturer and provide insights into the practical experiences made.
- We identify the shortcomings of Data Vault and demonstrate possible solutions.
- We present lessons learned and derive general success factors for the use of Data Vault in data lakes.
- We assess Data Vault as data modeling technique for structured data in data lakes.

The remainder of this paper is structured as follows: Section 2 describes the Data Vault model and its characteristics in detail. Sections 3 discusses the exemplary analytics projects, the difficulties that arose, and possible solutions. Section 4 assesses Data Vault based on the experiences made, presents the lessons learned, and derives success factors for Data Vault modeling. Section 5 gives an overview and comparative evaluation concerning modeling alternatives. Finally, Section 6 concludes the paper.

2 Data Vault Basics

After a first version of Data Vault [7], the current Data Vault model 2.0 extended and adapted the modeling technique further for enhanced performance [8]. Subsection 2.1 describes the Data Vault model's components and modeling guidelines. Subsection 2.2 details the key characteristics of Data Vault.

2.1 The Data Vault Model

This paper deals with Data Vault 2.0 as described in the reference book [8]. The Data Vault model is a conceptual and logical data model using table structures. Data Vault represents entities, relationships between entities, and additional context data in three different table types: *hubs*, *links*, and *satellites*.

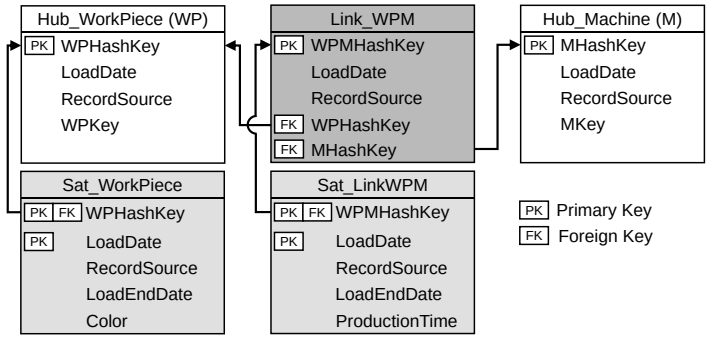

Fig. 1. The Data Vault model consists of three different table types: hubs, links, and satellites. Hubs represent business objects (e.g., work pieces or machines), while links connect hubs. Satellites contain descriptive attributes, such as *Color* or *ProductionTime* [8].

Hubs represent business objects in Data Vault. Two example hubs are depicted in white in Fig. 1. A hub contains the business key of the business object it represents, *WPKey* or *MKey* in the example, and a unique surrogate key *WPHashKey* or *MHash-Key*, hashed from this business key. Besides those two keys, a hub contains a load date, and a record source. The load date specifies when an entry was first added to the hub. The record source identifies the source system the entry was loaded from.

Links represent associations or hierarchies between hubs. They refer to the connected hubs via their surrogate keys. The primary key of each link is a hash of the business keys it connects. In the exemplary link in Fig. 1 (dark grey), the *WPMHashKey* is compound of the *WPKey* and the *MKey*. Like the hub, the link contains a load date and a record source. All links express many-to-many relationships. This adds flexibility to the model: If solely the cardinalities of a relationship in the source system change, this has no effect on the links in the Data Vault model. New entities can be added to the model without changing existing hubs, adding more flexibility. Instead, a new link is created or existing links are updated. There exist multiple types of links, e.g., *SAME-AS-links* indicate that entries of two hubs refer to the same business object.

Satellites add additional information to hubs and links. In the example in Fig. 1, the satellites (light grey) contain the color of a work piece or the production time associated with a combination of work piece and a machine. One hub or link may have multiple satellites holding all attributes that are not covered by the hubs and links themselves. Satellites contain the surrogate key of the hub or link they belong to as both primary key and foreign key. One satellite can hold multiple entries for the same hub or link entry for historization. Thus, the load date is the second part of the primary key, to create a unique identifier. In addition, each satellite entry contains the record source and a load end date. This load end date indicates when the entry's validity expired. Whenever data changes in the source system, a new entry is added to the satellite and the load end date of the old entry is updated. In this way, a historization according to Kimball's slowly changing dimensions type 2 [14] is achieved. Just like for links, there exist different types for satellites as well, such as *multi-active satellites*, which store multiple entries for one parent key (e.g., multiple phone numbers for one customer).

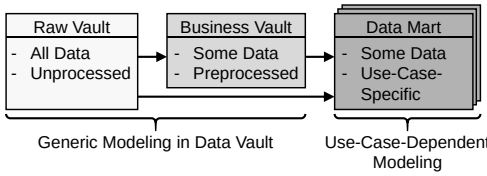

Fig. 2. Data Vault comprises two main parts: Raw Vault and Business Vault. The Raw Vault contains all data, while the Business Vault adds additional information to some data to increase performance. Use-case-specific Data Marts can be derived from either Vault [8].

Not all additional information on a hub or link should be kept in one single satellite, as this might lead to huge satellites. Instead, two splitting techniques are proposed: First, we may split satellites by source system. This eases the process of adding new source systems, as this only requires an additional satellite. Second, we may split satellites by the change frequency of contained attributes. With this, static attributes do not need to be updated every time a frequently changing attribute is changed.

In addition to these modeling structures, Data Vault comes with a *schema architecture* shown in Fig. 2. This architecture consists of the *Raw Vault*, the *Business Vault*, and *Data Marts*. Data first is loaded into the *Raw Vault*. In the Raw Vault, only *hard business rules* [9, 16] are applied, i.e., technical rules that do not change the meaning of data, such as the distribution into hubs, links, and satellites, or conversion into Unicode. Further transformations are applied in the *Business Vault*. Here, *soft business rules* [9, 16] are applied, which add business logic to the data. They might aggregate data, calculate KPIs, and much more. They may also add structures to improve performance, such as *bridge tables* containing frequently queried relationships. The Business Vault is an optional modeling layer based on top of a Raw Vault. It is not necessary to add all data from the Raw Vault to the Business Vault in the same level of detail. Finally the use-case-specific *Data Marts*, derived from Raw Vault or Business Vault, may be in any format, e.g., star schema.

2.2 Key Characteristics of Data Vault

The popularity of the Data Vault model is based on three key characteristics that result from its table structure: *flexibility*, *loading efficiency*, and *auditability* [7, 8].

Flexibility covers two aspects: (I) data are not changed in their meaning when saved in the Raw Vault. Instead, they are transferred into new tables and only hard business rules are applied. This means that they can be used for any desired use case. (II) the Data Vault model is easily adaptable and extendible. Changes in the source systems can easily be reflected in the Data Vault model with no or only little updates to existing tables [8]. Links are only updated if a new hub is added to the relationship. The addition of an attribute may be realized by updating a satellite, or by adding an entirely new satellite. In all other cases, such as adding a new entity or relationship, or even an entire source system, it is sufficient to add new tables to the Data Vault model. This supports

an agile approach in which one use case is implemented after the other and new business objects, relationships, and attributes are added on purpose. Data Vault 2.0 provides a project management methodology taking advantage of this characteristic [8].

The Data Vault model enables high *loading efficiency*. In Data Vault 1.0, tables of the same type could be loaded in parallel. However, the dependencies between tables enforce a certain order: first hubs, then links and finally satellites. Data Vault 2.0 addresses this issue, allowing all tables to be loaded in parallel.

The Data Vault model also provides *auditability*, as all changes made to a source system entry are stored in the satellites. For this, each change to the data is stored as a separate record with a timestamp that indicates its expiration date.

3 Data Vault Modeling for Data Lakes in Practice

Based on the key characteristics presented in Subsection 2.2, research literature suggests Data Vault to model data in data lakes [5]. To assess the suitability of this approach, we examined the usage of Data Vault in a real-world enterprise-wide data lake. This data lake is part of the industry 4.0 initiative of a large, global manufacturer, producing goods for various sectors, e.g., mobility or industry. Its data sources range from Enterprise Resource Planning (ERP) systems and Manufacturing Execution Systems (MES) to internet of things (IoT) devices.

We investigated the use of Data Vault in analytical projects from various business domains in the manufacturer's enterprise. In the following subsections, we detail three of them that provide significant insights into Data Vault modeling for data lakes. These business domains are *manufacturing* (Subsection 3.1), *finance* (Subsection 3.2), and *customer service* (Subsection 3.3). We identify ways in which the domains benefit from Data Vault, and present issues that arose and their possible solutions. Table 1 summarizes the characteristics of those domains. For the used data, we distinguish two data categories: (I) *Transactional enterprise data* that refer to business transactions and business objects, and (II) *non-transactional enterprise data* that originate from novel data sources (e.g., sensors or user generated content) and describe certain aspects of a business activity in detail. Other aspects of interest are the source systems involved, the process type [16], involved users, and addressed analytic capabilities.

3.1 Manufacturing Domain

In the manufacturing business domain, the goal of projects is to enable data-driven manufacturing [17]. The captured data are used for, e.g., process performance reporting and predictive maintenance. All projects are managed in an agile manner.

The focus of the analytics projects in this domain mainly lies on the analysis of *non-transactional data*. Data originate from numerous *MES* and are captured by sensors

Table 1. Overview over characteristics of the investigated domains.

	Manufacturing Domain	Finance Domain	Customer Service Domain
Used Data	Transactional, Non-transactional	Transactional	Transactional, Non-transactional
Kinds of Source Systems	ERP Systems, Master Data, MES, Manually added	ERP Systems	ERP Systems, IoT Devices, Master Data, Simulations
Process Type	Primary	Support	Primary
Involved Users	Business User, Domain Experts, Data Scientists	Business User, Domain Experts, Data Scientists	Domain Experts
Analytic Capabilities	Descriptive, Diagnostic, Predictive	Descriptive, Predictive	Descriptive

during manufacturing. *Transactional data*, such as master data, data from *ERP systems*, and *manually added data*, e.g., defect codes added by domain experts, are used as additional source of information. Since goods for sale are produced in this domain, the processes are *primary processes.* All kinds of users are equally involved in this analytics project. *Business users* create reports on different aspects of the process, such as factory efficiency (*descriptive* use cases). *Domain experts* use the data for *diagnostic* use cases, e.g., to analyze test results and optimize processes. *Data scientists* finally enable *predictive* use cases, such as predictive maintenance and quality assessment.

Experiences with Data Vault. The analytics projects in this domain benefit from the Data Vault characteristics in three ways: (I) the flexibility of Data Vault allows use-case-independent modeling, (II) facilitates the agile development, and (III) allows the incremental integration of numerous source systems, which is necessary due to the large number of different source systems involved in this domain.

One major issue arose during the usage of Data Vault in this business domain: Due to the integration of a large number of source systems the hash key generation became quite complex. Across all the different source systems, the same business key is often used for different business objects. In such a case, the Data Vault modeling reference suggests to either extend the business key with the source system, or to create one separate hub per source system. However, in the first approach, source systems are not properly integrated. The second approach would quickly result in a large and overly complex data model due to the large number of source systems involved in this domain (over 600). Thus, two different approaches to solve this issue were developed. In the first approach, the business key was extended using additional attributes to create a unique composed key. However, deciding which attributes to add is complex, especially when schemata or business logic change in the source systems, or when many business objects share the same values for a majority of their attributes. Therefore, a

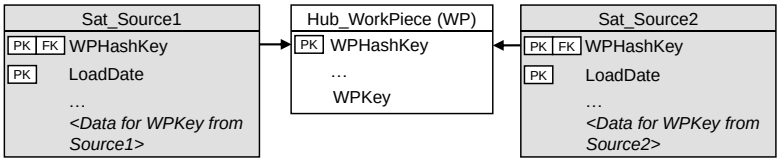

Fig. 3. If the same business key refers to different business objects in different source systems, one satellite per source system is added to the business key.

second solution was developed. In this approach, a satellite is added for each involved source system. If a business key is available in more than one source system, entries in all affected satellites are added. To retrieve the information on one certain object, both business key and source system have to be provided in the query. Fig. 3 shows an example hub with two associated source system satellites. This solution is similar to the proposed extension of the business key, but also integrates the different source systems.

Overall, even though the Data Vault reference did not sufficiently cover the issue of ambiguous business keys and thus had to be extended, Data Vault provides the significant benefits of flexibility and simple integration of new source systems. Especially regarding the large number of MES, this is an essential feature in this domain.

3.2 Finance Domain

The second business domain under consideration concerns finance and controlling. Data are used, for instance, to generate reports on wins and losses or to predict future revenue. Multiple teams work agilely and in parallel on independent projects in this domain. For example, one team is responsible for all use cases related to key performance indicator (KPI) calculation, while another team deals with prediction use cases. Only *transactional data* from *ERP systems* are used in this domain. The goal is to organize and coordinate other processes in the company, making this domain's processes *support processes*. Mainly *business users* are involved in this domain. They carry out *descriptive* use cases. However, *domain experts* and *data scientists* are also involved, focusing on *predictive use cases*, such as resource planning and revenue forecasts.

Experiences with Data Vault. In this business domain, the Data Vault characteristics benefit analytics projects in four ways: (I) Data Vault's flexibility allows use-case-independent modeling and (II) supports multiple teams working in parallel. (III) Data Vault's high loading efficiency makes source data quickly available for analysis, which is especially important for generating reports including recent data updates. (IV) Data Vault's auditability allows to detect tampering with sensitive data.

However, during the project iterations already carried out, the analytics project team encountered three difficulties affecting the Data Vault model and the modeling process:

A major issue was the *application of business logic*. The necessary business logic is split across the different layers of Data Vault (see Fig. 2): hard business rules are applied in the Raw Vault, while soft business rules are applied in the Business Vault. However, some business rules, e.g., currency conversion or resolving factorization, can not be clearly classified as hard or soft. For these rules, it is debatable whether the

Fig. 4. Modeling roles in relationships in Data Vault should be done using one link per role.

meaning of the data remains the same. Therefore, the project team extended the Data Vault reference to apply all these non-classifiable rules in the Raw Vault.

Modeling roles, i.e., the function an entity has in a relationship, in the Data Vault model is not covered by the Data Vault reference. However, roles are important in many use cases. For example, an order might have both a billing address and a shipping address, which have to be differentiated using roles. Again, two different approaches were evaluated. The first approach uses one link between the hubs involved in a relationship, i.e., the order and the address hub in the example. To this link, a satellite is added that contains the role of the relation. However, the link's primary key consists of the business keys of the related hub entries. If the billing address is equal to the shipping address for one certain order, two entries with the same primary key are added to the link table. The second and preferred approach therefore uses one link per role, as shown in Fig. 4.

To *save analytical results* for future use, e.g., to compare predictions to reality, an adaption of the Data Vault model is required. However, it is unclear how these results can be integrated into Data Vault, as this use case is not described in the Data Vault modeling reference. As a solution, a new hub was introduced that represents the analysis itself. It is linked to all hubs used in the analysis. Its satellites contain the analysis results. This way, source system data is clearly separated from processed data.

In summary, various issues arose in this domain that could not be solved using the standard Data Vault modeling reference. Instead, Data Vault had to be adapted or extended. However, the domain also greatly benefits from the key characteristics of Data Vault such as the flexibility for agile project management and parallel developer teams.

3.3 Customer Service Domain

The last domain to detail is the domain of customer service. Here, field data captured by IoT devices are used for, e.g., maintenance and product lifecycle management. For this purpose, a product is equipped with sensors that continuously capture data on its behavior. This field data then is compared to previously calculated simulation results.

The majority of data used are *non-transactional* field data captured by sensors and *simulation data*. *Transactional data*, in particular *master data* and data from *ERP systems*, are used to add additional information, such as product information. Since the data are used to improve the product and add value to the customer, the processes in this domain are *primary processes*. The only involved *user group* are *domain experts*, who execute *descriptive analysis* on the data, such as comparing field data to simulation data. Analytics projects in this domain are executed agilely.

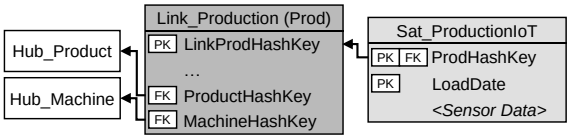

Fig. 5. To store sensor data, a nonhistorized link is suggested by the Data Vault modeling reference [8]. The associated satellite contains no load end date.

Experiences with Data Vault. Data Vault provides two major benefits to analytics projects in this domain through its flexibility, which (I) allows use-case-independent modeling and (II) supports agile project management.

The major issue encountered in this domain concerns the management of IoT data in Data Vault. Once captured, IoT data do not change. Thus, the Data Vault modeling reference suggests using *nonhistorized links* to store sensor data [8] (see Fig. 5). In such a link, the load end date is omitted. However, this quickly results in satellites with a large amount of entries due to the periodic capture of IoT data. Another idea developed in this domain was to store IoT data in an external system providing cheap storage for large amounts of data, e.g., HDFS. A link to these systems is stored in a satellite, similarly to link-based integration [18]. Using multi-active satellites [8], it is even possible to link multiple files to the same link or hub entry. However, this approach may lead to longer execution times, as the IoT data have to be retrieved from this external system.

Up until now, the project team has not yet found out which solution to prefer. Both approaches still have to be balanced against each other. However, it is clear that there are ways to manage even IoT data in the Data Vault model.

4 Lessons Learned and Overall Assessment

After examining the issues that arise in different business domains when using Data Vault in data lakes, this section deduces the lessons learned. To this end, we discuss and classify the issues encountered with Data Vault and highlight the solutions developed in the examined domains (Subsection 4.1). Based on these insights, we assess the adequacy of Data Vault for structured data in data lakes and derive generally valid success factors for Data Vault modeling (Subsection 4.2).

4.1 Lessons Learned and Classification of Issues

The issues that arose during the practical usage of Data Vault can be assigned to one of two classes (see Fig. 6): (I) issues only insufficiently covered by the modeling reference, and (II) issues not covered by the modeling reference at all.
(I) Insufficiently Covered:

- The *management of ambiguous business keys*. In this issue, the same business key refers to different business objects in various source systems (Section 3.1). The Data Vault modeling reference suggests two different approaches to model ambiguous

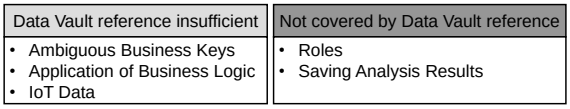

Fig. 6. Different issues arose when using Data Vault in data lakes, which the Data Vault modeling reference either only insufficiently covered or not covered at all.

business keys, which both have their drawbacks. Instead, one hub was created and separate satellites for the different source systems were added.
- The *application of business logic* (Section 3.2). In the finance domain, not all rules could clearly be classified as hard or soft. In the domain, the developers classified these ambiguous business rules as hard rules and applied them in the Raw Vault.
- The *handling of IoT data* (Section 3.3). The Data Vault modeling reference suggests storing IoT data in a nonhistorized link. However, the satellite tables containing these data would become very large. Alternatively, IoT data can be stored in an external low-cost system. So far, there is no evidence on which approach is preferable.

(II) Not Covered:

- The *modeling of roles in relationships* (Section 3.2). To solve this issue, one link per role was added between the affected hubs.
- The *saving of analysis results* (Section 3.2). In this case, an additional hub was added to the model to represent the analysis itself and to contain the results.

Both of these uncovered issues are not specific for the analyzed domains. Roles in relationships occur in multiple business divisions, e.g., in human resources to differentiate the roles of people working on a certain project. Storing analysis results or transformed data back into the data lake is also a reoccurring use case [19, 20]. The solutions developed for these issues comply with Data Vault modeling by using only existing modeling structures (links and satellites). However, an extension to the Data Vault model to directly cover these issues would be worthwhile to have.

4.2 Assessment and Success Factors for Data Vault in Data Lakes

Overall, analytics projects in each of the three examined business domains profited from the key characteristics of Data Vault mentioned in Section 2.2. Especially Data Vault's flexibility was beneficial in all domains, due to agile project management, integration of multiple source systems, and support of parallel development. The business domains also benefitted from high loading efficiency and auditability. However, various issues arose during modeling. From the experience gained with these issues, we derive three generally valid success factors for Data Vault modeling in data lakes:

(I) Identify shortcomings of the Data Vault modeling reference. As shown above, not all issues encountered in the projects are sufficiently covered by the Data Vault reference, requiring an extension or adaption of the reference.

(II) Define a data architecture for data lakes. While for data warehouses, the Data Vault reference proposes a schema architecture (see Fig. 2) [8], such an architecture does not exist for data lakes. It is unclear whether the given architecture is applicable in the data lake context. However, defining the Data Vault layers and the applied business rules is of great importance and thus could be the basis of a data lake architecture.

(III) Identify inconsistencies in source systems. These issues, such as ambiguous business keys, may lead to severe problems in integration and analysis. Therefore, they have to be addressed during data integration.

These success factors necessitate an enterprise-wide set of data modeling guidelines that contain both modeling specifications and best practices for data modeling. Thereby, these guidelines extend or even change the Data Vault modeling reference to fit the context. They also should be communicated across domains to ensure consistent data modeling across the data lake. We conclude that combined with such guidelines, Data Vault is well suited to model structured data in data lakes.

5 Related Work and Comparative Evaluation

While we discussed Data Vault for data lake modeling, there are other alternatives from both the data warehouse and the data lake context. Subsection 5.1 presents these alternate modeling techniques. Subsection 5.2 compares Data Vault to some of these alternatives using criteria relevant in the studied domains and data lakes in general.

5.1 Related Work

Representing the real world as accurately as possible is the aim of the well-known entity-relationship model (ER model) [21]. Here, business objects are modeled as entities with relationships between them. However, the ER model is only a conceptual model and other techniques are used for logical and physical modeling.

For the data warehousing context, dimensional modeling was developed as conceptual and logical model [14]. Data is stored in either fact tables or dimension tables. Fact tables contain the metrics and measurements of interest for the business, e.g., sales figures. Dimension tables allow to aggregate these so-called facts, e.g. along a time axis.

Another approach to data warehouse modeling is normalization [22]. Especially the third normal form was used for logical modeling. To allow historization, the third normal form can be alternated into so called head-version tables [9]. Here, attributes are divided into static attributes and attributes that should be historized. Static attributes are stored in a so-called head table together with the business key. Attributes to be historized are stored in one or more version tables linking to the respective head table.

As a next step, Data Vault emerged as combination of dimensional modeling and third normal form [7]. It is a conceptual and logical modeling technique.

The digitalization poses new challenges on data analytics and data management, which are addressed in data lakes [2, 3]. For this context, additional modeling techniques were recently proposed. Data droplets, for example, model the entire data lake as a composition of small RDF graphs [23]. In another modeling technique [24], each

Table 2. Comparative evaluation of modeling techniques

	Data Vault	Dim. Modeling	Head-Version Tables
Use-Case-Independence	Yes	No	Yes
Support of agile Project Management	Yes	No, many changes necessary	Often, adaption is necessary
Source Schema Changes	Few changes necessary	Mostly big changes necessary	Mostly big changes necessary
High Loading Efficiency	Yes	Parallel loading of dimensions possible	No
Auditability	Yes	Yes	Yes
Number of Tables	Very large	Small	Large
Query Performance	Many JOINS	Few JOINS	Depends
Understandability	Medium	Very high	Medium
Integration of Multiple Source Systems	Very simple using satellites	Complex	Complex

data entry is modeled as a small graph of four nodes, containing different information on the entry, e.g., the data itself or its metadata. These four-tuples then are connected to each other via their metadata nodes. To our knowledge, there exists no practical experience report on the adequacy of these modeling techniques for data lakes.

5.2 Comparative Evaluation

As shown in the course of this paper, different domains with different requirements benefitted from the use of Data Vault for data lake modeling. Nevertheless, Data Vault also revealed some weaknesses in the form of insufficiently addressed modeling issues. We thus compare Data Vault to dimensional modeling [14] and head-version tables [9] (as an alternation of third normal form) to evaluate whether these alternatives are more suitable for data lake modeling, using criteria relevant in the domains and data lakes in general. We will not investigate ER modeling, as it is only conceptual, nor first modeling approaches developed for data lakes specifically (such as data droplets [23]), as these modeling approaches are still immature and not widespread in practice.

Table 2 depicts the result of this qualitative comparison. *Use-Case-Independence*, as is necessary in data lakes, is achieved by all techniques but dimensional modeling, where the analytic goals define the model. *Support of agile project management* (see investigated domains) is only provided by Data Vault. In case of the other modeling techniques, the schema of already existing tables has to be changed, e.g., to add a new attribute. Similarly, *Source Schema Changes*, which happen in especially agile projects, result in many changes for dimensional modeling and head-version tables but not for

Data Vault. *High Loading Efficiency*, as needed in the domains, is provided by Data Vault and partially by dimensional modeling, where dimension tables can be loaded in parallel. In head-version tables, too many dependencies between tables make parallel loading very complex. *Auditability* is achieved in all techniques using e.g., slowly changing dimensions for dimensional modeling [14]. The *Number of Tables* is small for dimensional modeling, where one fact table and only few tables for dimensions are needed. Data Vault typically has an even higher number of tables than the head-version model, especially in cases, where many one-to-one relationships are involved. While head-version tables represent these relationships without additional tables, Data Vault creates a link table for each of them. The *Query Performance* is directly dependent of the number of tables, which is why dimensional modeling typically needs fewer JOINS than Data Vault. The *Understandability*, which plays a role whenever non-data scientists use the data, is affected by the number of tables as well, but also by the overall complexity of the model. Here, dimensional modeling is easiest to understand due to its simple structure. The *Integration of Multiple Source Systems*, as needed in the domains, finally is simple in Data Vault and can be solved using satellites. In the other techniques however, more complex integration techniques are necessary.

Overall, Data Vault addresses most criteria very well. However, especially dimensional modeling has its strengths were Data Vault has its weaknesses (number of tables, performance, and understandability). Thus, in cases where these criteria are of great importance, e.g., for KPI-focused or aggregation-focused use cases, we propose to use dimensional data marts on top of Data Vault, as already indicated in Fig. 2.

6 Conclusion

Data lakes recently emerged to enable the use-case-independent use of data. However, even data in a data lake have to be modeled. Without data modeling, data are prone to quality and integration issues. Research literature suggests Data Vault for this purpose. To determine the adequacy of Data Vault for data lake modeling, we examined real-world business domains at a large, globally active manufacturer. We provided insights into three domains and discussed the experiences made with the practical application of Data Vault for data lakes. It turned out that even though some of the projects used data rather untypical for Data Vault (e.g., IoT data), it was successfully applied in all projects. However, multiple issues arose when using Data Vault, some that were only insufficiently covered by the Data Vault modeling reference, some that were not covered at all. To successfully use Data Vault in data lakes, a set of enterprise-wide modeling guidelines is necessary, which extend the available Data Vault modeling reference and contain solution approaches and best practices.

References

1. Margulies, J.C.: Data as Competitive Advantage. Winterberry Gr. (October), 1–28 (2015).
2. Mathis, C.: Data Lakes. Datenbank-Spektrum. 17 (3), 289–293 (2017).

3. Fang, H.: Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem. In: Proc. of the 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER 2015). (2015).
4. Russom, P.: Data Lakes - Purposes, Practices, Patterns, and Platforms. TDWI. (2017).
5. Topchyan, A.R.: Enabling Data Driven Projects for a Modern Enterprise. Proc. Inst. Syst. Program. RAS (ISP RAS 2016). 28 (3), 209–230 (2016).
6. Stiglich, P.: Data Modeling in the Age of Big Data. Bus. Intell. J. 19 (4), 17–22 (2014).
7. Linstedt, D.: Super Charge Your Data Warehouse: Invaluable Data Modeling Rules to Implement Your Data Vault. (2012).
8. Linstedt, D., Olschimke, M.: Building a Scalable Data Warehouse with Data Vault 2.0. Elsevier LTD (2015).
9. Schnider, D., Martino, A., Eschermann, M.: Comparison of Data Modeling Methods for a Core Data Warehouse. Trivadis. (2014).
10. Yessad, L., Labiod, A.: Comparative study of data warehouses modeling approaches: Inmon, Kimball and Data Vault. In: 2016 International Conference on System Reliability and Science (ICSRS). (2016).
11. Jovanovic, V., Bojicic, I.: Conceptual Data Vault Model. In: Proc. of the 15th Southern Association for Information Systems Conference (SAIS 2012) (2012).
12. Krneta, D., Jovanovic, V., Marjanovic, Z.: A Direct Approach to Physical Data Vault Design. Comput. Sci. Inf. Syst. 11 (2), 569–599 (2014).
13. Cernjeka, K., Jaksic, D., Jovanovic, V.: NoSQL document store translation to data vault based EDW. In: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO 2018). (2018).
14. Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. Wiley (2013).
15. Inmon, W.H., Linstedt, D.: Data Architecture: A Primer for the Data Scientist - Big Data, Data Warehouse and Data Vault. Elsevier LTD (2014).
16. Porter, M.E.: Competitive Advantage: Creating and Sustaining Superior Performance. Free Press (1985).
17. Gröger, C.: Building an Industry 4.0 Analytics Platform. Datenbank-Spektrum. 18 (1), 5–14 (2018).
18. Gröger, C., Schwarz, H., Mitschang, B.: The Deep Data Warehouse: Link-Based Integration and Enrichment of Warehouse Data and Unstructured Content. In: Proc. of the 2014 IEEE 18th International Enterprise Distributed Object Computing Conference (EDOC 2014). (2014).
19. IBM Analytics: The governed data lake approach. IBM. (2016).
20. Terrizzano, I., Schwarz, P., Roth, M., Colino, J.E.: Data Wrangling: The Challenging Journey from the Wild to the Lake. In: Proc. of the 7th Biennial Conference on Innovative Data Systems Research (CIDR'15) (2015).
21. Chen, P.P.-S.: The Entity-Relationship Model-Toward a Unified View of Data. ACM Trans. Database Syst. 1 (1), 9–36 (1976).
22. Inmon, W.H.: Building the Data Warehouse. Wiley (2005).
23. Houle, P.: Data Lakes, Data Ponds, and Data Droplets, http://ontology2.com/the-book/data-lakes-ponds-and-droplets.html, (2017).
24. Walker, C., Alrehamy, H.: Personal Data Lake with Data Gravity Pull. In: Proc. of the 2015 IEEE Fifth International Conference on Big Data and Cloud Computing (BDCloud'15). IEEE (2015).