Universität Rostock Traditio et Innovatio

WIND WIRTSCHAFTSINFORMATIK UNIVERSITÄT ROSTOCK

Varianten zur Speicherung multi-dimensionaler Daten

Qualitative und quantitative Bewertung von Star- und Snowflake-Schema, Column Stores und Data Vault

eingereicht am 05.04.2016

von

Robert Schiller | Lüneburger Straße 13 | 18057 Rostock

Matrikel-Nr.: 209204212

Gutachter:

PD Dr.-Ing. habil. Meike Klettke

Universität Rostock

Albert-Einstein-Str. 22

18059 Rostock

Zweitgutachter:

Prof. Dr.-Ing. habil. Peter Forbrig

Universität Rostock

Albert-Einstein-Str. 22

18059 Rostock

UNIVERSITÄT ROSTOCK - FAKULTÄT FÜR INFORMATIK UND ELEKTROTECHNIK

LEHRSTUHL FÜR DATENBANK- UND INFORMATIONSSYSTEME

D-18051 ROSTOCK

Danksagung

Mein Dank geht an dieser Stelle an Alle, die mich im Laufe meiner Studienzeit unterstützt, gefördert und gefordert haben. Ein ganz besonderer Dank geht an meine Betreuerin Frau PD Dr.-Ing. habil. Meike Klettke, die mich in der Pflichtveranstaltung „Data Warehouses/Data Mining“ für diese Thematik begeistern konnte und ohne die diese Masterarbeit sicherlich nie entstanden wäre. Mit Ruhe, Geduld und Verständnis stand sie mir bei all meinen Problemen, die im Laufe der Arbeit auftraten, zur Seite. Danke!

Ein ganz besonderer Dank gilt auch Luisa Neumann, die meine Stimmungsschwankungen die kompletten 6 Monate ohne Murren ertragen hat und immer wieder als Motivator an meiner Seite stand. Danke!

Nicht zuletzt danke ich auch meiner Familie und speziell meinen Eltern, die mir es letztlich doch immer ermöglicht haben, dass ich meinen Weg gehen konnte. Auch in schwierigen Zeiten konnte ich mir eurer Unterstützung immer sicher sein und ich weiß, dass ich es eigentlich viel zu selten sage: Danke!

Kurzfassung

Data Warehouse Systeme haben sich als wichtiger Bestandteil der strategischen Unternehmensplanung etabliert. Die zentralen Lager dienen vor allem den Wissensarbeitern eines Unternehmens als Grundbaustein für Analyseverfahren, um im Hinblick auf strategische und taktische Ausrichtungen des Unternehmens eine Entscheidungsfindung zu unterstützen.

Mit Data Vault 2.0 ist eine neue komplexe architektonische Erweiterung für Data Warehouses entstanden, die nicht nur eine Referenzarchitektur, sondern auch einen neuen Ansatz eines Datenmodells, eine agile Methodologie und Best Practices für die Implementation beinhaltet. Neben dem Konzept paralleler Architekturen, schließt Data Vault auch Thematiken rund um Big Data, Echtzeit-Laden und der Integration unstrukturierter Daten mit ein.

Im Rahmen dieser Masterarbeit steht vor allem die Analyse des Data Vault Datenmodells im Mittelpunkt, welches mit den traditionellen Speichervarianten des Star- und Snowflake-Schemas sowie den Column Stores auf Vor- und Nachteile verglichen wird. Zum einen wird ein qualitativer Vergleich auf Basis verschiedener Anforderungen an Data Warehouses durchgeführt. Zum anderen wird durch einen quantitativen Vergleich die Analyse der Datenmodelle um messbare Kriterien erweitert. Um dabei eine objektive Vergleichbarkeit zu schaffen, werden zum quantitativen Vergleich der TPC-H Benchmark als auch der Starschema Benchmark herangezogen.

Schlagwörter: Data Vault, Data Warehouse, Starschema, Snowflake-Schema, Column Stores, TPC-H Benchmark, Starschema Benchmark

Abstract

Data warehouse systems are well established in strategic business planning. They act as central units for huge datasets and are especially used by knowledge workers from an analysis point of view in order to support an alignment on strategical and tactical activities in a business environment.

Featuring Data Vault 2.0, a new complex extension has been developed regarding data warehouse systems that not only includes a reference architecture but also a new approach for a conceptual data model, an agile methodology and Best Practices for implementation issues. In addition to the concept of parallel architectures, it also implies topics like Big Data, realtime loading and the integration of unstructured data.

Within the scope of this master thesis the new Data Vault 2.0 is analyzed and compared to the traditional data warehouse models: Star schema, Snowflake schema and Column Stores. On the one hand a qualitative comparison is done by analyzing typical data warehouse requirements. On the other hand a quantitative comparison regarding performance issues is performed in order to evaluate advantages and disadvantages of a Data Vault model. On that account the TPC-H Benchmark as well as the Star Schema Benchmark are considered to guarantee an objective comparability.

Key words: Data Vault, Data Warehouse, Star schema, Snowflake schema, Column Stores, TPC-H Benchmark, Star Schema Benchmark

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Im betrieblichen Umfeld gewinnt die adäquate Verwaltung großer Datenbestände einhergehend mit deren Analyse seit mehreren Jahren zusehends an Bedeutung. Die Grundlage dabei bilden zentrale Datenlager, die auch als Data Warehouses bezeichnet werden. Im Gegensatz zu den tagtäglichen Prozessen (z.B. der Erweiterung des Kundenbestands) in operativen Systemen, welche der Aufrechterhaltung des fortlaufenden Betriebs gelten, liegt das Hauptaugenmerk in Data Warehouses auf dem Sammeln, Aufbereiten, Abfragen, Auswerten und Bereitstellen möglichst aller Datenbestände jener operativen Systeme. Dabei dient es vor allem den Wissensarbeitern eines Unternehmens (Manager, Projektleiter, etc.) als Grundbaustein für Analyseverfahren, um im Hinblick auf strategische und taktische Ausrichtungen des Unternehmens eine Entscheidungsfindung zu unterstützen.

Um den Zweck eines Data Warehouses zu ermöglichen, wurde auf logischer Ebene das multidimensionale Datenmodell eingeführt. Dieses spezielle Datenmodell ermöglicht die Darstellung von für Analysen notwendigen betrieblichen Kennzahlen aus unterschiedlichen kontextbezogenen Perspektiven. [Kimball et al. 2011; Goeken 2007; Chaudhuri et al. 1997; Lusti 1998; Lusti 1998]

Bei der Umsetzung des logischen Datenmodells in physische Datenstrukturen haben sich verschiedene Speicherverfahren für multidimensionale Daten, die in Data Warehouses zugrunde liegen, etabliert. Unter anderem bilden relationale Schemata, wie das Star- und Snowflake-Schema, und Column Stores die Basis verschiedener Data Warehouse Systeme. Während das Starschema durch genau eine Faktentabelle und jeweils eine denormalisierte Tabelle pro Dimension modelliert wird und somit nur in der 1. Normalform existiert, ist mit dem Snowflake-Schema ein Modell entwickelt worden, dass die einzelnen Klassifikationshierarchien in unterschiedlichen Tabellen modelliert und somit die dritte Normalform umsetzt. Da zugunsten der beiden Schemata entweder Performanz- oder Redundanzvorteile überwiegen, wurde mit dem Column Stores ein gänzlich anderer Ansatz entwickelt, bei dem eine spaltenweise Speicherung der Tupel erfolgt und somit Vorteile bei der Anfragebearbeitung entstehen. [Köppen et al. 2014]

Mit Data Vault 2.0 ist nun ein neues Speicherverfahren im Umlauf, welches die Hub- und Spoke-Architektur, die 2. und 3. Normalform relationaler Datenbankmodellierung und zum Teil die dimensionale Modellierung, repräsentiert durch das Starschema, miteinbezieht. Hohe Flexibilität bei Erweiterungen und eine starke Parallelisierung bei Datenladeprozessen unterstützen die Agilität von Data Warehouses. [Köppen et al. 2014; Linstedt et al. 2015]

Damit existieren drei verschiedene Klassen von Speicherungsverfahren für multidimensionale Anwendungen, die im weiteren Verlauf dieser Masterarbeit qualitativ und quantitativ verglichen

werden. Ergebnis der Arbeit soll ein Kriterienkatalog für den qualitativen Vergleich sein, die Ergebnisse des qualitativen Vergleiches, die Ergebnisse des quantitativen Vergleiches mittels Benchmarks sowie deren Analyse, um über das Potenzial einer Data Vault Architektur bezüglich deren Akzeptanz und Etablierung in der Data Warehouse Landschaft zielführende Aussagen treffen zu können.

Bevor ein Kriterienkatalog anhand von Literatur aus der Datenbank- und Data Warehouse Theorie in Abschnitt 4 erstellt wird, werden die Speicherungsverfahren im Einzelnen, illustriert an einem einfachen Beispiel, erklärt. Auf dessen Grundlage erfolgt dann sowohl ein erster quantitativer Vergleich durch eine eigens dafür implementierte Testumgebung in Abschnitt 5, als auch der qualitative Vergleich in Abschnitt 6. Um eine objektive Vergleichbarkeit zu schaffen, wird in Abschnitt 8 ein weiterer quantitativer Vergleich durchgeführt, der sowohl auf Grundlage vom TPC-H Benchmark als auch vom Starschema Benchmark erstellt und realisiert wird.

2. Multidimensionale Speicherung

Multidimensionale Daten können unter anderem in relationalen Datenbankmanagementsystemen gespeichert werden. Diese Art der Speicherung basiert auf dem multidimensionalen Datenmodell, welches die Grundlage für analytische Systeme bildet. Die wesentlichen Elemente des Modells werden in Abschnitt 2.1 beschrieben. Zur Illustration der einzelnen Merkmale wird ein einfaches Beispiel herangezogen, welches innerhalb dieser Arbeit immer wieder zur Verdeutlichung einzelner Thematiken herangezogen wird.

Aufbauend darauf werden in Abschnitt 2.2 sowohl die Varianten der zeilenorientierten Speicherung für eine Abbildung auf relationalen Strukturen, das heißt auf Tabellen, als auch die Variante der Column Stores beschrieben.

2.1 Das multidimensionale Datenmodell

Das multidimensionale Datenmodell beinhaltet Kennzahlen, die aus verschiedenen Perspektiven (Dimensionen) analysiert werden. Abbildung 1 illustriert dieses spezielle logische Datenmodell. Dabei bilden Produkt, Zeitraum und Region die einzelnen Dimensionen, welche mögliche Sichten auf Kennzahlen beschreiben. Hierfür ist es oft notwendig, Dimensionen in Konsolidierungsebenen zu unterteilen, um Hierarchien darstellen zu können, mit deren Hilfe man verdichtete und verfeinerte Kennzahlen erhalten kann. In Abbildung 1 wäre solch eine Hierarchie bei der Dimension „Region“: Filiale → Stadt → Bundesland. Je nach Ausprägung in den Dimensionen kann der Wert einer Kennzahl bestimmt werden. Somit bilden die einzelnen Dimensionswerte die Koordinaten im mehrdimensionalen Raum. [Köppen et al. 2014]

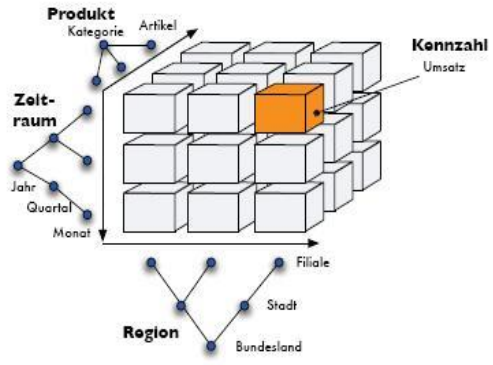


Abbildung 1: Multidimensionales Datenmodell (Auszug aus [Köppen et al. 2014])

Die Zellen des Datenmodells bilden die Kennzahlen. In Abbildung 1 wäre eine Kennzahl der Umsatz, welcher in einer bestimmten Filiale, in einem bestimmten Monat, von einem bestimmten Artikel, entstanden generiert wurde. Kennzahlen sind numerische Größen, die größtenteils kon-

solidiert, also verdichtet sind. Sie beschreiben größtenteils betriebswirtschaftliche Zusammenhänge, wie den Umsatz. [Goeken 2007; Köppen et al. 2014]

In einem multidimensionalen Datenmodell muss jede Dimension eine endliche Menge von mindestens zwei Dimensionselementen aufweisen. Hat ein Datenmodell 2 Dimensionen, kann es als Tabelle dargestellt werden. Hat man drei Dimensionen, wie in Abbildung 1, entsteht ein Würfel. Da man sich nicht mehr als drei Dimensionen bildlich vorstellen kann, findet man in der Literatur auch öfter den Begriff „Data Cube“ bzw „Datenwürfel“ für ein multidimensionales Datenmodell. [Köppen et al. 2014; Goeken 2007]

2.2 Multidimensionale Speichervarianten

Um ein multidimensionales Modell in einem relationalen Datenbankmanagementsystem abbilden zu können, müssen gewisse Bedingungen eingehalten werden. Vor allem die anwendungsbezogene Semantik des multidimensionalen Modells sollte verlustfrei umgesetzt werden. Darüber hinaus muss es möglich sein, multidimensionale Anfragen unter Berücksichtigung der Anfragecharakteristik und des Datenvolumens von Analyseanwendungen in relationale Anfragen umzusetzen und effizient auszuführen. [Köppen et al. 2014]

Es gibt verschiedene Schemata, die eine zeilenorientierte Speicherung gewährleisten. Zu den wohl bekanntesten gehört neben dem Starschema auch das Snowflake-Schema. Eine andere Variante bilden die Column Stores, die sich anhand ihrer spaltenorientierten Speicherung charakterisieren lassen. Im Folgenden werden die drei Varianten genauer betrachtet.

2.2.1 Starschema

Das Starschema basiert auf genau einer Faktentabelle und mehreren Dimensionstabellen, wobei pro Dimension jeweils nur eine Dimensionstabelle mit einem eindeutig identifizierbaren Primärattribut und allen Klassifikationsattributen existiert. Abbildung 2 illustriert das Starschema anhand eines Beispiels, angelehnt an den in Abschnitt 2.1 beschriebenen Datenwürfel. Hinzugekommen ist die Dimension „Kunde“, die aufgrund der vereinfachten dreidimensionalen Darstellung des Modells als Datenwürfel in Abbildung 1 noch nicht vorkommt.

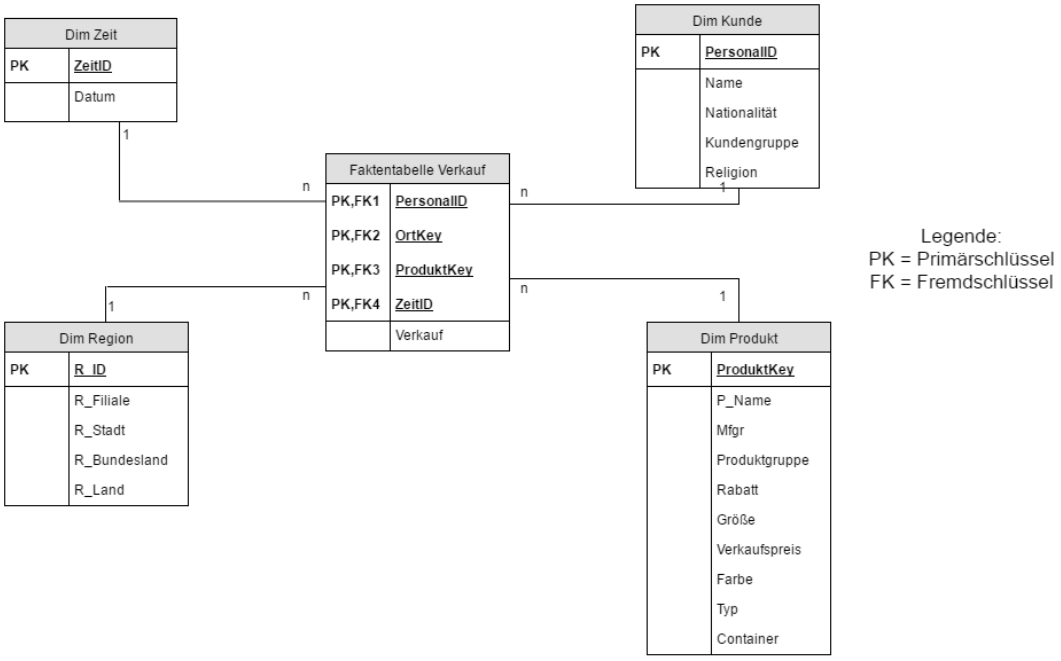


Abbildung 2: Beispiel Starschema

Die einzelnen Dimensionen „Kunde“, „Produkt“, „Region“ und „Zeit“ besitzen, wie schon oben beschrieben, jeweils ein Primärattribut. In diesem Falle sind es die im Schema unterstrichenen und durch den Zusatz „PK“ gekennzeichneten Keys und IDs: PersonalID, ProduktKey, R_ID und ZeitID. Weiterhin ist deutlich zu erkennen, dass die einzelnen dargestellten Konsolidierungsebenen einer Dimension aus Abbildung 1 in ein und derselben Tabelle existieren. So besitzt zum Beispiel die Tabelle „Produkt“ auch das Attribut „Produktgruppe“, welches letztendlich eine andere Klassifikationsstufe dieser Dimension beschreibt. Die Tatsache, dass die Klassifikationshierarchien einer Dimension aufgelöst und in einer einzelnen Tabelle dargestellt werden, hat zur Folge, dass die zu einer Dimension gehörende Tabelle denormalisiert ist und es somit zu Updateanomalien und Redundanzen kommen kann. [Köppen et al. 2014; Levene et al. 2003]

Die Faktentabelle dagegen enthält Kennzahlen und Events von Geschäftsprozessen oder anderen unternehmerischen Bereichen, die üblicherweise metrisch sind, um analytische Verfahren darauf anwenden zu können. Im Beispiel besitzt die Faktentabelle „Verkauf“ somit die Kennzahl „Umsatz“, welche sich in diesem Fall aus den Attributen „Rabatt“ und „Verkaufspreis“ der Dimension „Produkt“ errechnen lässt. Weiterhin enthält jede Relation Fremdschlüssel, die jeweils auf ein Schlüsselattribut der Dimensionstabelle verweisen, sodass der Tabelle „Verkauf“ die Fremdschlüssel „PersonalID“, „R_ID“, „ProduktKey“ und „ZeitID“ angehören, die auch gleichzeitig den zusammengesetzten Primärschlüssel der Faktentabelle bilden. [Köppen et al. 2014]

Aufgrund der Denormalisierung und der damit einhergehenden Möglichkeit von Updateanomalien und Redundanzen, wird mit dem Snowflake-Schema ein anderer Ansatz verfolgt.

2.2.2 Snowflake-Schema

Auch wenn der Aufbau auf Grundlage des multidimensionalen Datenmodells, bestehend aus Kennzahlen und deren Betrachtung aus verschiedenen Perspektiven, gleichbleibt, verhält es sich doch anders mit dem Snowflake-Schema. Wie in Abbildung 3 zu sehen ist, unterstützt das Snowflake-Schema Attributhierarchien, indem es einem erlaubt „Subdimension"-Tabellen zu erstellen, sodass jede Hierarchiestufe jeweils eine eigene Tabelle erhält, welche durch einen Fremdschlüssel auf die nächst höhere Stufe referenziert. Als Beispiel kann wieder die Dimension „Produkt" herangezogen werden, bei der die Klassifikationsstufe „Produktgruppe" nun nicht wie beim Starschema in einer einzigen Dimensionstabelle vorhanden ist (siehe Abbildung 2), sondern als eigenständige Entität in der Tabelle „Produktgruppe" gespeichert wird. [Köppen et al. 2014; Levene et al. 2003]

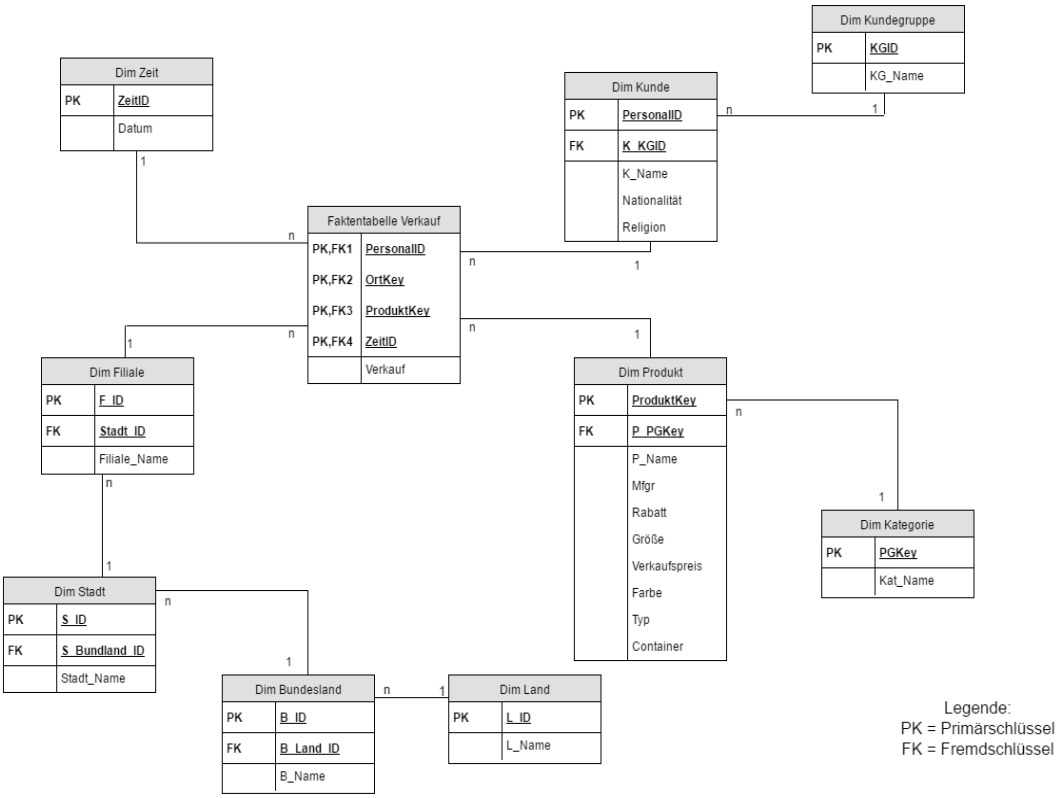


Abbildung 3: Beispiel Snowflake-Schema

Die Faktentabelle beinhaltet neben den Kennzahlen und Events nur noch die Fremdschlüssel der Klassifikationsstufe mit der niedrigsten Granularität. Das Aufspalten der Attributhierarchien in einzelne Tabellen ermöglicht eine exakte Implementation in der 3NF und somit auch Redundanzfreiheit und die Verringerung von Updateanomalien. Im Gegensatz zum Starschema ist die Bearbeitungsdauer größerer Anfragen wesentlich langsamer, da viele indirekte Verknüpfungen zwi-

schen den einzelnen Tabellen vonnöten sind. [Levene et al. 2003; Linstedt et al. 2015; Köppen et al. 2014]

Für welche Variante sich Unternehmen entscheiden oder ob man eine Mischform wählt, hängt von verschiedenen Entscheidungskriterien ab. Dazu zählen zum Beispiel die Änderungshäufigkeit der Dimensionen, die Anzahl der Dimensionselemente und die Anzahl der Klassifikationsstufen einer Dimension. Es bedarf immer einer Betrachtung der Charakteristika von Data Warehouse Anwendungen, um die Vor- und Nachteile der einzelnen Schemata abzuwiegen. [Köppen et al. 2014]

2.2.3 Column Stores

Das Prinzip der Column Stores beschreibt eine weitere Form der Speicherung, bei der das spaltenweise Speichern im Vordergrund steht. Im Gegensatz zur zeilenorientierten Speicherung, bei welcher jedes einzelne Tupel eines relationalen Datenbankmanagementsystems (DBMS) immer auch intern zusammenhängend gespeichert wird, nimmt man bei den Column Stores ein spaltenorientiertes Zusammenfügen nach den Attributen vor. Bei dieser Variante ist zu beachten, dass einzelne Tupel auch immer wieder rekonstruiert werden können. Die Rekonstruierbarkeit lässt sich durch zwei verschiedene Verfahren gewährleisten. [Abadi et al. 2009]

Einerseits kann beim Speichern eine feste Tupelreihenfolge vordefiniert sein, so dass in allen Attributlisten dieselbe Reihenfolge genutzt werden kann. Als Referenz auf diese Listen dient dann zum Beispiel eine stabile Liste aus Tupelidentifikatoren. Andererseits ist es möglich eine vertikale Partitionierung mit Hilfe von künstlichen Surrogatschlüsseln vorzunehmen, wodurch eine Key-Value-Speicherung erfolgt, bei dem jedem Attribut eines Tupels der gleiche Surrogatschlüssel zugeordnet wird. Diese Art der Speicherung ist im Hinblick auf die Speicherplatzoptimierung jedoch nicht so effektiv wie die erste Variante, da für jedes Attribut die volle Liste der Surrogatschlüssel gespeichert werden muss. Dennoch ermöglicht die daraus entstehende Dekomposition, abgeleitet von dem DSM (Decomposed Storage Model), eine volle Nutzung der relationalen Technologien, wie zum Beispiel das Definieren von Indexen. [Abadi et al. 2009; Köppen et al. 2014]

Die Vorteile von Column Stores gegenüber anderen relationalen Speichervarianten liegen klar bei der Anfragebearbeitung auf einzelne Spalten und bei der Optimierung des Speicherverbrauchs. Aufgrund der großen Wahrscheinlichkeit von sowohl syntaktischer (gleicher Datentyp) als auch semantischer (gleiche Eigenschaften) Homogenität von Werten eines Attributs, können Duplikate erkannt und komprimiert werden. Es gibt diverse Komprimierungsverfahren, beispielsweise die Run-Length-Encoding- oder Dictionary-Methode, die durch bestimmte Vor- und Nachteile ver-

schiedenartige Daten behandeln und verarbeiten können. Nachteile besitzen die Column Stores vor allem beim Arbeiten auf vielen Spalten, wie dem Schreiben neuer Tupel. [Abadi et al. 2009]

3. Data Vault

Data Vault 2.0 zeichnet sich durch hohe Flexibilität bei Erweiterungen und eine starke Parallelisierung bei Datenladeprozessen aus und unterstützt somit die Agilität von Data Warehouses. Das Ziel ist es daher die komplexe Logik, die einem Data Warehouse zugrunde liegt, soweit wie möglich zum Endnutzer hin zu bewegen, damit Veränderungen von Datenquellen, Datenstrukturen etc. schnellstmöglich adaptiert werden können. [Linstedt et al. 2015]

Data Vault 2.0 ist somit nicht nur eine Variante zum Speichern multidimensionaler Daten, es bietet auch eine Lösung für ein komplettes Business-Intelligence-System an. Während sich die erste Version des Data Vault Ansatzes lediglich auf die physischen und logischen Datenmodelle fokussierte, kommen mit Data Vault 2.0 weitere Komponenten hinzu. Neben einer Methodologie werden auch eine Referenzarchitektur beschrieben und Implementationsvorschläge gegeben, die auf Grundlage industrieweiter Best Practices entwickelt wurden. Auch das Data Vault Modell aus der ersten Version wurde zur Performance-Steigerung und besseren Skalierbarkeit angepasst. [Linstedt et al. 2015]

Um die Unterschiede zu den in Abschnitt 2.2 beschriebenen multidimensionalen Speicherverfahren und die Besonderheiten des Data Vault Ansatzes herauszufiltern, wird im Folgenden vor allem auf die Data Vault Architektur in Abschnitt 3.1 und auf das Data Vault Datenmodell in Abschnitt 3.2 eingegangen.

3.1 Data Vault Architektur

Die typische Data Warehouse Architektur nach Kimball besteht aus zwei Ebenen: ETL-Schicht und Data Warehouse Schicht. Nachdem in der ETL-Schicht die Rohdaten aus den Datenquellen extrahiert, bereinigt und in die Data Warehouse Schicht geladen werden, werden in der Data Warehouse-Schicht, die nach der Logik des multidimensionalen Datenmodells aus Abschnitt 2.1 modelliert ist, die bereinigten Daten in Data Marts für den Endnutzer bereitgestellt. [Kimball et al. 2011]

Die Data Vault Architektur jedoch basiert auf der von Inmon entwickelten Architektur eines atomischen Data Warehouses. [Inmon et al. 1994] Dementsprechend besteht es, im Gegensatz zu den traditionellen DWH-Architekturen, aus drei Ebenen: Staging Area, Enterprise Data Warehouse Layer (EDW-Layer) und Information Delivery Layer. (Abbildung 4)

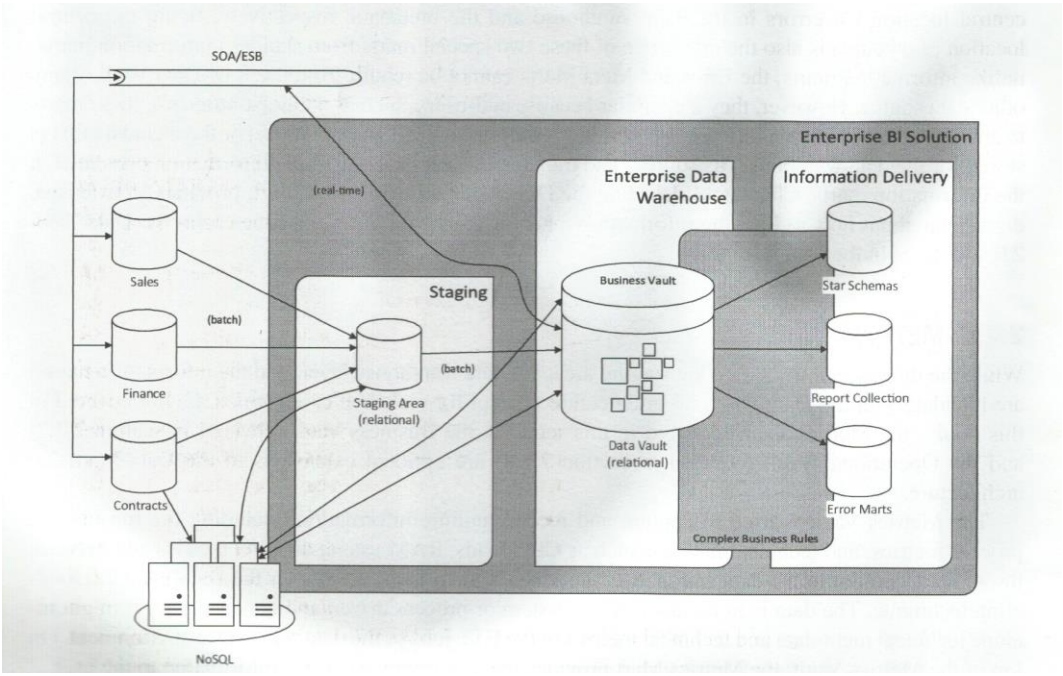


Abbildung 4: Data Vault Architektur (Auszug aus [Linstedt et al. 2015, S. 28])

In der Staging Area werden, wie bei der ETL-Schicht im typischen Data Warehouse, die Rohdaten aus diversen Datenquellen extrahiert und dies so schnell wie möglich, um den Workload auf die operativen Systeme zu reduzieren. Der Unterschied zur ETL-Schicht besteht darin, dass keine historischen Daten in der Staging Area lagern, sondern nur der Stapel (Batch) an Daten, der in die nächste Schicht, dem EDW-Layer, geladen werden soll. Dies hat zur Folge, dass man nicht mit Strukturveränderungen von Daten zu kämpfen hat, was eine gewisse Logik in den Datenladeprozessen mit sich ziehen und dem Ziel von Data Vault 2.0, die Logik möglichst weit zum End-User hin zu implementieren, widersprechen würde. [Linstedt et al. 2015]

Der EDW-Layer beinhaltet alle historischen, zeitbezogenen Daten, jedoch immer noch als Rohdaten, also unbearbeitete Daten, die in der Granularität gespeichert werden, in der sie in den Datenquellen vorkommen. Außerdem unterstützt der EDW-Layer sowohl unstrukturierte Daten als auch das Echtzeitladen mittels Service Oriented Architecture oder Enterprise Service Bus. Ein weiterer Unterschied zu den traditionellen Data Warehouse Architekturen besteht darin, dass der Endnutzer keinen Zugriff auf den Data Warehouse Layer hat, sondern nur auf den Information Mart Layer. [Linstedt et al. 2015]

Der Information Mart Layer stellt die für den Endnutzer nützlichen Daten in Data Marts bereit. Diese Data Marts werden meist als Starschema modelliert und durch dahinterliegende Logik themenorientiert und in aggregierter Form für Zwecke des Reportings bereitgestellt. Verlangt der Endnutzer jedoch ein Data Mart basierend auf der 3NF im Snowflake-Schema, können die Data Marts auch auf dessen Bedürfnisse angepasst werden, da der Endnutzer nur den Information

Mart Layer erreicht und der EDW-Layer transparent für ihn ist. Optional zum einfachen EDW-Layer bestehend aus den historisierten Rohdaten, kann der EDW-Layer auch noch „at top“ ein Business Vault besitzen, wenn die eingesetzte Logik zwischen dem einfachen EDW-Layer (Raw Vault) und dem Information Mart Layer zu komplex wird. Auf das Business Vault werden dann schon gewisse Regeln angewendet und Daten in aggregierter Form gespeichert. Somit kann das Laden in die Information Marts erleichtert werden. [Linstedt et al. 2015]

Durch den Vergleich der Data Vault Architektur mit der traditionellen Architektur nach Kimball, kommt man zu der Erkenntnis, dass die dem DWH zugrundeliegende Logik soweit wie möglich zum Endnutzer hin verschoben wurde. Im Gegensatz zu den traditionellen DWHs wird der Großteil aller Regeln, die im DWH implementiert sind, beim Erstellen der Information Marts eingesetzt, wenn auch ein Teil schon in der Staging Area angewendet wird. Somit gilt es, zwischen zwei Arten von Regeln zu unterscheiden. Einerseits gibt es die „Hard Rules“, welche sich nur um technische Details kümmern und die Rohdaten nicht verändern. Das heißt, sie sind syntaktischer Herkunft und werden beim Laden von Daten in die Staging Area angewendet. Es gibt unterschiedliche Typen von Hard Rules, um zum Beispiel Restriktionen zu definieren, die angeben, welche Daten aus den Quellen geladen werden dürfen. Weitere Hard Rules beziehen sich auf das Verhalten des Systems bei bestimmten Situationen, wie fehlende Daten, oder auch auf die Art und Weise, wie Fehler dokumentiert werden sollen. Auf der anderen Seite stehen die „Soft Rules“, welche die semantische Ebene darstellen und sich vor allem der Modifikation der Rohdaten annehmen, um sie für die Endnutzer effektiv bereitzustellen. Zum Beispiel können Neukalkulationen oder Neuinterpretationen von bestimmten Rohdaten geregelt werden. Die Umsetzung der Soft Rules nahe dem Endnutzer führt, wie oben schon beschrieben, zu einer schnelleren Adaption veränderter Datenquellen, da eventuell nur syntaktische Regeln angepasst werden müssen. [Linstedt et al. 2015]

3.2 Das Data Vault Modell

Data Vault Modelle orientieren sich an natürlichen, komplexen Netzwerken, wie sie überall auf der Welt zu finden sind. Dabei soll eine nahezu vollständige Umsetzung eines natürlichen Modells in ein businesszentriertes Modell, unter Berücksichtigung diverser kritischer Charakteristika, die ein Business beschreiben, vollzogen werden. Das Hauptaugenmerk liegt dabei auf den Business Keys. Business Keys sind Schlüssel zur Beschreibung, wie einzelne Unternehmen Informationen aufnehmen, integrieren, verbinden und anfragen können. Sie sollten daher eineindeutig und nicht veränderbar sein. Beispielsweise sind Produktnummern (z.B. ISBN) oder Kundennummern zu nennen. [Linstedt et al. 2015]

Anhand der oben genannten Eigenschaften erfolgt eine Umsetzung des Modells aus Abbildung 2 mit Hilfe von drei Komponenten: Hubs, Links und Satelliten.

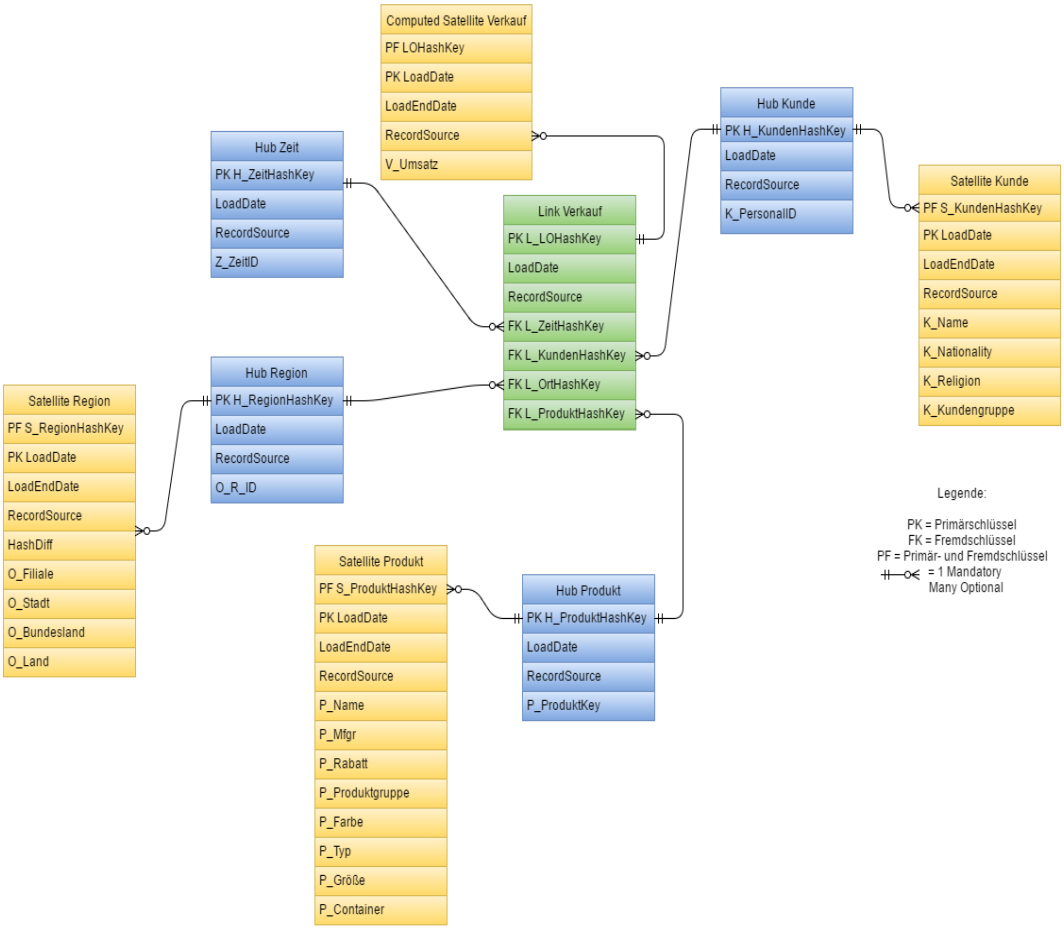


Abbildung 5: Bestandteile eines Data Vault Modells

Abbildung 5 stellt die Bestandteile eines Data Vault Modells anhand des vom Autor in Abbildung 2 entwickelten Beispiels mit den Entitäten „Kunde“, „Produkt“, „Zeit“ und „Region“ dar. Im Folgenden werden die einzelnen Komponenten beschrieben.

3.2.1 Hubs

Hubs bilden die zentralen Komponenten in einem Data Vault Modell. Da den Business Keys eine große Bedeutung durch deren Identifizierung von Business Objekten gegeben wird, werden sie vom Rest des Modells getrennt und separat gespeichert. Die Aufgabe eines Hubs besteht demnach darin, einen oder mehrere Business Keys eines Objekts und weitere Informationen zu diesen zu speichern. Abbildung 6 zeigt anhand des Hubs „Hub Produkt“ die einzelnen Attribute, die diese Komponente besitzen muss.

Abbildung 6: Beispiel Hub

Diese weiteren Informationen, auch Metadaten genannt, enthalten Informationen über den Business Key. Zum einen wird immer mit Hilfe des Attributs „RecordSource“ beschrieben, auf welche Datenquelle sich der Business Key bezieht, beziehungsweise, aus welcher Datenquelle der Business Key geladen wurde. Zum anderen wird für den in das Data Vault geladenen Batch ein vom System erstellter Zeitstempel in Form des Attributs „LoadDate“ generiert. Mit Hilfe des Zeitstempels können zum Beispiel Fehler und technische Probleme beim Ladeprozess schneller identifiziert werden. Außerdem wird für jeden Hub ein Hash generiert, der den Primärschlüssel des Hubs darstellt. Der Hash-Schlüssel wird aus den im Hub gespeicherten Business Keys gebildet und referenziert als Fremdschlüssel zu anderen Link- und Satellit-Entitäten. Da die meisten Unternehmen entweder intelligente Schlüssel, d.h. zusammengesetzte Schlüssel, generiert aus Schlüsseln anderer Business Objekte (z.B. der EAN Barcode), oder Surrogat-Schlüssel, also künstlich erzeugte Schlüssel, als Business Keys nutzen, soll mit Hilfe des Hash-Schlüssels eine Performance-Verbesserung erzielt werden, da diese Schlüssel eine konstante Länge aufweisen, während die Business Keys unterschiedlicher Objekte meist eine variable Länge besitzen und der Lookup auf diesen Schlüsseln tendenziell langsamer ist. Ein anderer Grund für die Generierung der Hash-Schlüssel besteht darin, dass dadurch eine einfachere Integration von NOSQL-Datenbanken erfolgen kann, da die dahinterliegenden Systeme, wie zum Beispiel Hadoop, mit Hash-Schlüsseln arbeiten. Optional kann dem Hub auch noch das Attribut „LastSeenDate“ innewohnen, welches nur dann benutzt wird, wenn die Datenquellen kein Change Data Capture (CDC) Konzept anwenden, also selbst das Kennzeichnen modifizierter Daten übernehmen. Dies ist bei den meisten operativen Datenbanksystemen der Fall, sodass mit Hilfe des LastSeenDate das letzte Datum angegeben wird, an welchem der Business Key in der Datenquelle gesehen wurde. Wird eine festgelegte Zeitspanne, in der der Business Key in der Datenquelle nicht mehr gesehen wurde, überschritten, kann mit Hilfe dieses Attributs der Datensatz als in der Datenquelle gelöscht betrachtet werden. [Linstedt et al. 2015]

Zusammengefasst besteht ein Hub dementsprechend aus den folgenden Attributen:

- Hash Key
- Business Key

- Load Date
- Record Source
- Last Seen Date (optional)

3.2.2 Links

Links dienen der Hierarchie-, Transaktions- und Assoziationsabbildung zwischen den Hubs. Abbildung 5 zeigt eine Assoziation zwischen allen Entitäten anhand des Links „Link Verkauf“. Die Verbindungen, die dadurch entstehen, beziehen sich auf vergangene, gegenwärtige und zukünftige Beziehungen von Business Keys. Implementiert durch Tabellen existieren im Data Vault Modell grundsätzlich nur Many-to-Many Beziehungen durch Link-Entitäten, welche allerdings auch flexibel 1:m, n:m, m:1 und 1:1 Beziehungen umsetzen können. Dadurch kann die Stabilität auch bei Änderungen von Kardinalitäten gewährleistet werden. Während die ETL-Ladeprozesse in traditionellen Data Warehouses umgeschrieben werden müssen, sobald sich die Definition einer Beziehung zwischen Entitäten ändert, wird dies durch die flexible Handhabung der Links im Data Vault Modell nicht verlangt, da Hubs und Links beliebig hinzugefügt werden können, ohne die bereits vorhandenen Komponenten zu beeinflussen. Die Granularität eines Links wird durch die Anzahl der Hubs, die er verbindet, bestimmt. Je mehr Hubs ein Link verbindet, desto feiner ist seine Granularität. Das selbe Verhalten findet sich beim Starschema wieder. Wenn dort eine neue Dimension zur Faktentabelle hinzugefügt wird, verringert sich auch die Granularität der Fakten. [Linstedt et al. 2015]

Ein Link weist eine ähnliche Struktur wie die eines Hubs auf. Er besteht aus:

- Hash Key
- Load Date
- Record Source
- Hash Key vom referenzierten Hub
- Last Seen Date (optional)
- Dependent Child Key (optional)

Abbildung 7 illustriert die einzelnen Komponenten anhand des Links „Link Verkauf“. Da dieser Link eine Beziehung zwischen allen Hubs wiederspiegelt, beinhaltet er alle Hash-Schlüssel der einzelnen Hubs als Fremdschlüssel. Weiterhin werden auch hier die Metadaten „LoadDate“ und „RecordSource“ aufgenommen, die, wie bei den Hubs, den Ladezeitpunkt und die Datenquelle angeben. Neben dem optionalen „LastSeenDate“, dessen Funktion in Abschnitt 3.2.1 beschrieben wurde, ist auch das optionale Attribut „Dependent Child Key“ in dem Link aus Abbildung 7 nicht vorhanden. Ein „Dependend Child Key“-Attribut gibt den Referenzschlüssel zu Entitäten an,

die durch Parent-Child Hierarchien modelliert wurden. Weiterhin besitzt auch jeder Link einen Hash-Schlüssel, der als Primärschlüssel dient und aus den Hash-Schlüsseln der Hubs errechnet wird. Dass Links einen eigenen Hash-Schlüssel besitzen, hat den Vorteil, dass so die von der Staging Area in das Data Warehouse zu ladenden Daten direkt über den Hash-Schlüssel und nicht durch weiterführende Joins zu den referenzierten Hubs mit den bereits im Link vorhandenen Daten verglichen werden müssen. [Linstedt et al. 2015]

Abbildung 7: Beispiel Link

3.2.3 Satelliten

Um den Business Objekten eine Bedeutung zu geben, werden Satelliten implementiert. In Abbildung 5 wird dem Hub „Hub Produkt“ ein Satellit „Satellit Produkt“ zugeordnet. Satelliten beinhalten alle Daten, die der Beschreibung eines Objekts dienen. Da der Kontext eines Objekts variieren kann und veränderbar ist, muss ein Satellit in der Lage sein diese Veränderungen zu tracken und umzusetzen. Ein Satellit ist immer genau einem Hub oder Link zugeordnet. Strukturell ist auch ein Satellit ähnlich aufgebaut wie ein Hub. In Abbildung 8 wird deutlich, dass Satelliten neben den Metadaten die ein Business Objekt beschreibenden Elemente enthält und somit aus einer Vielzahl an Attributen bestehen kann. [Linstedt et al. 2015]
Ein Satellit besteht aus folgenden Attributen:

- Parent Hash Key
- Load Date
- Record Source
- Load End Date
- Extract Date (optional)
- Hash Diff (optional)
- Beschreibende Attribute

Der Parent Hash Key ist Teil des zusammengesetzten Primärschlüssels eines Satelliten und dient gleichzeitig als Referenz zum Primärschlüssel eines Hubs oder Links. Dadurch besteht eine Verbindung zwischen dem Business Key eines Business Objekts und den beschreibenden Attri-

buten. Weiterhin besitzt jeder Satellit auch einen Ladezeitstempel und einen Eintrag über die Datenquelle. Im Gegensatz zu den Hubs und Links kommt noch ein Load End Date hinzu, das einen Satelliten zu einer endlichen Komponente macht und einen Eintrag nach einer gewissen Zeitspanne für ungültig erklären kann. Es ist außerdem das einzige Attribut, das im Satelliten verändert werden kann. Optional kann einerseits ein Extract Date hinzugefügt werden, welches die Zeit des Extrahierens eines Datensatzes aus der Datenquelle erfasst, um historische Ladeprozesse zu verstehen. Andererseits kann optional auch noch ein Attribut „Hash Diff“ hinzukommen, das zum Vergleich von bereits vorhandenen Datensätzen mit neu hinzukommenden benutzt wird. Es ist daher der Zweckmäßigkeit des Link Hash-Schlüssels ähnlich. Der Hashwert des Attributs wird aus allen beschreibenden Attributen bereits in der Staging Area errechnet und eingefügt. Wird nun ein neuer Datensatz in das Data Vault geladen, dessen Business Key bereits vorhanden ist, werden die beiden Hashwerte des Attributs „Hash Diff“ vom alten und neuen Eintrag miteinander verglichen, um zu sehen, ob sich bei den beschreibenden Attributen Veränderungen ergeben haben. Dieses Attribut hat zum Vorteil, dass nicht über alle anderen Attribute einzelne Vergleiche laufen müssen, um eventuelle Veränderungen zu identifizieren. [Linstedt et al. 2015]

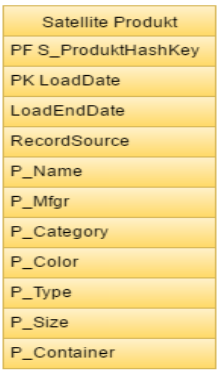


Abbildung 8: Beispiel Satellit

Die beschreibenden Attribute sollten auf verschiedene Satelliten aufgeteilt werden, um die Performance bei Ladeprozessen und die Flexibilität bei Aktualisierungsprozessen zu verbessern. Einerseits wird vorgeschlagen, dass für jede Datenquelle ein eigenständiger Satellit modelliert werden soll, der dann die Attribute in der für ihn ausgewiesenen Datenquelle tracked. Außerdem sollten die Satelliten auch auf Grundlage der Wahrscheinlichkeit des Auftretens von Attributveränderungen erstellt werden. In anderen Worten, es sollte jeweils einen Satelliten geben, der die Attribute enthält, dessen Werte sich kaum ändern (z.B. Name, Vorname, Religion bei einem Kunden), eventuell einen Satelliten mit Attributen, deren Änderungswahrscheinlichkeit eher mittlerer

Natur ist (z.B. Adresse, Telefonnummer) und einen Satelliten, der die Attribute enthält, die sich häufig aktualisieren (z.B. Kontostand). [Linstedt et al. 2015]

Neben den eben beschriebenen optimierten Satelliten können je nach Bedarf jedoch auch andere Arten von Satelliten modelliert werden:

Overloaded Satellites sind Satelliten, die die beschreibenden Attribute aus allen Datenquellen speichern. Somit entstehen multiple Einträge pro Business Key, die sich anhand von LoadDate und RecordSource unterscheiden.

Multi-active Satellites sind Satelliten, die multiple Einträge pro Business Key speichern, allerdings von ein und derselben Datenquelle. Diese Satelliten entstehen durch denormalisierte Datenquellen, wie XML-Files. Beispiel: Ein Kunde hat mehrere Nationalitäten. Da diese Daten alle zeitgleich in das Data Warehouse geladen werden und sich nur in den Nationalitäten unterscheiden, muss noch ein weiterer Primärschlüssel hinzugefügt werden, um eine eindeutige Identifizierung zu ermöglichen. Dieser Primärschlüssel kann zum Beispiel eine Sequenznummer sein, der die doppelten Einträge sequenziert.

Status Tracking Satellites werden genutzt, um Audit Trails oder Daten von CDC-Systemen zu laden.

Effectivity Satellites werden meist an Links angebunden, um die Effektivität derer zu ermitteln. Dabei werden die im Link in Beziehung stehenden Business Objekte zum Beispiel anhand ihrer Business Keys getrackt, um dann herauszufinden, ob beziehungsweise bis wann der Link noch aktiv ist.

Record Tracking Satellites werden eingesetzt, um Veränderungen in den Datenquellen zu ermitteln.

Computed Satellites errechnen und speichern Aggregationen aus vorhandenen Daten. Da im Raw Vault des EDW-Layers an sich nur die Rohdaten gespeichert werden, können diese Satelliten nur im Business Vault umgesetzt werden. Das in Abbildung 5 illustrierte Beispiel besitzt einen berechneten Satelliten „Computed Satellite Verkauf“, indem das Attribut V_Umsatz einen bereits berechneten Wert enthält. Computed Satellites werden im Grunde genommen genutzt, damit Daten, die mit hoher Wahrscheinlichkeit in berechneter Form für viele Data Marts interessant sind, nur einmal anhand von Business Rules berechnet werden, anstatt für jeden Data Mart einzeln. Dies führt zu erheblichen Performance-Verbesserungen. [Linstedt et al. 2015]

4. Kriterienkatalog

Die Implementation von sowohl operativen Datenbanken als auch analytischen Data Warehouses wird nach bestimmten Anforderungen und Kriterien umgesetzt. Um ein Kriterienkatalog für Data Warehouses zu entwickeln, wurde vorhandene Literatur aus dem Bereich der Datenbanktheorie und die der Data Warehouses durchsucht. Dabei lag der Fokus vor allem auf Regeln und Anforderungen, die beim Aufbau von Datenbanken und Data Warehouses beachtet werden sollten.

Zwischen operativen und analytischen Datenbanken gibt es neben dem unterschiedlichen Fokus (Lesen/Schreiben/Modifizieren/Löschen vs. Lesen/periodisches Hinzufügen) auch Unterschiede im Hinblick auf das Datenvolumen und der Anwenderzahl (sehr viele vs. sehr wenige). Auch der Anwendertyp unterscheidet sich. Während operative Datenbanken von Angestellten oder einer Applikationssoftware zur Ein- und Ausgabe verwendet werden, benutzen analytische Datenbanken vor allem Manager, Controller und Analysten. Die Anfragen bei operativen Datenbanken beziehen sich auf wenige Datensätze einer Datenquelle und vorrangig auf einen Zugriff einzelner Tupel, wohingegen in Data Warehouses durch komplexe Anfragen, analysebezogen und aus mehreren Datenquellen stammend, auf viele Datensätze spaltenweise zugegriffen wird. [Wrembel et al. 2007] [Köppenetal.É

Es wird ersichtlich, dass Data Warehouses eigenständige Systeme sind, die von operativen Datenbanksystemen differenziert betrachtet werden müssen. Daher werden im Folgenden vor allem Anforderungen für den Katalog betrachtet, die aus der Data Warehouse Literatur stammen.

Auf den ersten Blick war zu erkennen, dass es einerseits eher pragmatische Anforderungen an eine DWH-Architektur gibt, wie die Unabhängigkeit zwischen Datenquellen und Analysesystemen, die dauerhafte Bereitstellung integrierter und abgeleiteter Daten (Persistenz), die Mehrfachverwendbarkeit der bereitgestellten Daten, und die Durchführung prinzipiell beliebiger Auswertungen. [Köppen et al. 2014]

Weiterhin werden auch konkretere Anforderungen benannt, wie die Unterstützung individueller Sichten, die Erweiterbarkeit, die Automatisierung der Abläufe, und die Eindeutigkeit über Datenstrukturen, Zugriffsberechtigungen und Prozesse. [Köppen et al. 2014]

Inwiefern welche Anforderungen zu welchem Ausmaß umgesetzt werden, hängt auch von der Wahl des Speicherungsverfahrens ab. Nichtsdestotrotz gibt [Codd et al. 1993] einen ersten Anhaltspunkt für alle Data Warehouse Systeme, indem er 12 OLAP-Regeln (Online Analytical Processing-Regeln) definiert hat, die jedes System umsetzen sollte.

1. Multidimensionale Sicht
2. Transparenz

3. OLAP-Zugriffe
4. Performance
5. Skalierbarkeit
6. Generische Dimensionalität
7. Dünnbesetzte Strukturen
8. Mehrbenutzerbetrieb
9. Uneingeschränkte Operationen
10. Intuitive Benutzeroberfläche
11. Flexibles Reporting
12. Beliebig viele Dimensionen und Aggregationsebenen

Diese Kriterien sind nicht unumstritten, daher erfolgte eine Erweiterung um 6 Punkte: der Datenintegration, der Unterstützung unterschiedlicher Analysemodelle (kategorisches, exegetisches, kontemplatives, und formelbasiertes Modell), der Trennung der operativen Daten von Analysedaten, der Trennung der Speicherorte, der Unterscheidung zwischen Null- und Fehlerwerten und der Behandlung von Fehlerwerten. [Köppen et al. 2014]
Weitere Anhaltspunkte bieten [Pendse 1995], die die FASMI-Anforderungen definiert haben:

1. Fast: kurze Antwortzeiten
2. Analysis: adäquate Schnittstelle
3. Shared: mehrere Nutzer
4. Multidimensional: multidimensionales Modell
5. Information: vollständige Informationsbereitstellung

Alle bisher genannten Faktoren müssen in jedem Data Warehouse erfüllt sein, damit eine adäquate Nutzung erfolgen kann. Bezüglich des Anforderungskatalogs sollten auch „weiche Kriterien“ aufgenommen werden. Der Begriff kommt aus den Wirtschaftswissenschaften und beschreibt „nicht bzw. nicht objektiv quantifizierbare Größen, die Einfluss auf den Erfolg eines Unternehmens oder Projekts haben.“ [Peter et al. 1982] In dieser Arbeit werden weiche Faktoren als subjektiv wahrgenommene Unterschiede hinsichtlich der verschiedenen Speichervarianten definiert. Dazu zählen unter anderem die Benutzbarkeit, die Verständlichkeit von Abfragen und die Übersichtlichkeit von Datenmodellen.
Ein weiteres wichtiges Kriterium beschreibt die Performance und ihre Leistungsindikatoren. Die Performance hat im Allgemeinen zwei Bedeutungen. Zum einen beschreibt die Performance die Geschwindigkeit, in der ein System operiert, entweder theoretisch (Mtops = million theoretical operations per second) oder durch das Zählen von ausgeführten Operationen und Instruktionen

(MIPS = Million Instructions Per Second), innerhalb eines Benchmark-Tests. Zum anderen kann Performanz aber auch durch die totale Effektivität eines Systems definiert werden. Diese würde den Durchsatz, die individuelle Antwortzeit und die Verfügbarkeit beinhalten. [Margaret Rouse 2006]

Als Ergebnis kann man festlegen, dass der erweiterte OLAP-Katalog von Codd die Vielfalt an Anforderungen recht weit abdeckt. Lediglich die Beachtung von weichen Faktoren wie die Verständlichkeit und Einfachheit von Abfragen kommt eventuell zu kurz, auch wenn die Anforderung der intuitiven Benutzeroberfläche in dieselbe Richtung geht. Da die meisten Anforderungen sich auf ganze Data Warehouse Systeme beziehen, die Arbeit sich fortlaufend jedoch nur mit den im Data Warehouse Layer eingesetzten unterschiedlichen Datenmodellen befasst, kommen nur einige Kriterien für einen Vergleich infrage:

- Performance: Wie schnell wird die Ergebnismenge einer Abfrage generiert und korrekt ausgegeben?
- Abfragekomplexität: Sind die kontextbezogenen Fragestellungen des Anwenders einfach und verständlich in Abfragen umzusetzen?
- Uneingeschränkte Operationen: Werden alle Operationen, die dem Datenwürfelmodell zugeordnet sind, uneingeschränkt unterstützt?
- Multidimensionale Sicht: Ist die multidimensionale Sicht konzeptionell realisiert worden?
- Erweiterbarkeit des Datenmodells: Lässt sich das Datenmodell ohne große Probleme erweitern?
- Struktur des Datenmodells: Ist das Datenmodell übersichtlich und einfach gehalten?

Ein Data Warehouse Modell sollte demnach Grundlage einer guten Performance sein, für den Anwender einfach in der Anwendung, vor allem in Bezug auf die Definition von Abfragen, Operationen uneingeschränkt unterstützen und einfach erweiterbar sein. Des Weiteren sollte ein Data Warehouse Modell die multidimensionale Sicht unterstützen.

Für den quantitativen und qualitativen Vergleich zwischen Star- und Snowflake findet sich in der Literatur eine Vielzahl an Artikeln. Das Starschema ist das weit verbreitetste S für Data Warehouses, auch aufgrund seiner guten Performance gegenüber dem Snowflake-Schema. [Köppen et al. 2014; Srivastava et al. 2014] Aufgrund der Ergebnisse dieser Performance-Analysen wird das Snowflake-Schema beim quantitativen und qualitativen Vergleich in dieser Arbeit nicht betrachtet. Es findet somit nur ein Vergleich zwischen Starschema und Data Vault Schema statt, da die Performance als eines der wichtigsten Kriterien eingestuft wird. Auch [Pendse 1995] zieht in den FASMI-Anforderungen einen direkten Bezug zur Performance. Demnach sollten einfache Abfra-

gen maximal fünf Sekunden dauern, komplexe Abfragen maximal 20 Sekunden. Da die Performance durch Messwerte bestimmt wird, wird im folgenden Abschnitt 5 eine Testumgebung implementiert, auf deren Basis eine Messung stattfinden kann.

5. Quantitativer Datenmodell-Vergleich

Grundlage des Vergleichs quantitativer Kriterien ist der Aufbau einer Testumgebung, auf der das in Abschnitt 4 beschriebene Kriterium der Performance gemessen werden soll. Im Folgenden wird schrittweise erklärt, wie man solch eine Umgebung aufbauen und zum Vergleich der Datenmodelle nutzen kann. Die ersten beiden Abschnitte beschreiben dabei die Einrichtung der Testumgebung und Formulierung der Abfragen. Die Messung des Leistungsindikators erfolgt letztendlich in Abschnitt 5.3.

5.1 Einrichtung der Testumgebung

Die Testumgebung besteht, basierend auf den verschiedenen Speichervarianten, aus mehreren Datenbanken, welche über denselben semantischen Kontext verfügen, sich syntaktisch allerdings durch die für jede Speichervariante charakterisierenden Merkmale unterscheiden. Das Starschema basiert auf dem Beispiel aus Abbildung 2. Auf dessen Grundlage wird einerseits eine zeilenorientierte Speicherung und andererseits eine spaltenorientierte Speicherung durch Column Stores umgesetzt. Das Data Vault besteht aus den in Abschnitt 3.2 beschriebenen Komponenten. Um diese Datenbankmodelle zu implementieren, müssen folgende Punkte bedacht werden:

Datenbankmanagementsystem: Als Erstes muss das DBMS ausgesucht werden, auf dem die Testumgebung implementiert werden soll. Hierfür wurde der Microsoft SQL Server 2014 ausgewählt. Die Wahl fiel auf den SQL Server 2014, da dieser den strukturierten Spalten Index anbietet. Der eigentliche Gedanke, die Speicherung der Column Stores durch eine Key-Value-Speicherung simulieren zu lassen und darauf aufbauend den quantitativen Vergleich durchzuführen, wurde verworfen, da im Gegensatz zum SQL SERVER 2012 mit dem SQL SERVER 2014 nun die Möglichkeit besteht, nicht nur nicht-gruppierte Indizes für spaltenorientierte Tabellen zu definieren, sondern auch gruppierte Indizes (Structured Column Store Index). Gruppierte Indizes sind deshalb bei diesem Vergleich von Bedeutung, da komplette Tabellen spaltenorientiert abgespeichert werden können und somit eine wesentlich aufwendigere Umsetzung der Key-Value-Speicherung umgangen werden kann. Einziger Knackpunkt dieses Indizes ist, dass keine weiteren Indizes, zum Beispiel die der Schlüsselvergabe, darauf definiert werden können, um eventuelle Performance-Optimierungen vorzunehmen. Nichtsdestotrotz ist dies im Rahmen dieser Arbeit vernachlässigbar, da die spaltenorientierte Speicherung an sich mit der Data Vault Speicherung verglichen werden soll und nicht der Vergleich zwischen verschiedenen Indizes innerhalb dieser.

Auflistung der Hardware: Das Auflisten der verwendeten Hardware ist von Bedeutung, damit die erzielten Ergebnisse auch nachzuvollziehen sind, da es bei unterschiedlicher Hardware zu

unterschiedlichen Werten bei der Performance-Messung kommen kann. Die für den qualitativen Vergleich verwendete Hardware, auf dem die Datenbanksoftware SQL SERVER 2014 lief, war ein Lenovo Thinkpad Edge mit folgenden Komponenten:

- Mainboard: Lenovo 3259HLG
- Prozessor: Intel® Core™ i5-3210M CPU @ 2.50GHz, 2501 MHz, 2 Kern(e), 4 logische Prozessoren
- Arbeitsspeicher: 4GB (DDR3 SDRAM)
- Festplatte: 500 GB SATA II 3.0Gb/s (2.5") Festplatte mit 7.200 Umdrehungen pro Minute (upm)
- Solid State Drive: -
- Laufwerk: CD / DVD-ROM Laufwerk
- Netzteil: Original Lenovo 65 Watt 20 V Netzteil
- Betriebssystem: Microsoft Windows 10 Pro

Grund für diese Auswahl war die Tatsache, dass man auf diesen Server über den gesamten Zeitraum der Arbeitsphase uneingeschränkten Zugriff hatte. Da nicht die Server-Performance an sich, also die Geschwindigkeit des Rechners, im Vordergrund steht, sondern die einzelnen Vergleiche der Speichervarianten auf ein und demselben Server, können auch auf Basis dieser eingeschränkten Hardware solche Vergleiche durchgeführt werden. Auf dem Datenbankserver wurden alle Hintergrunddienste, die nicht vom Betriebssystem als notwendig angesehen wurden, wie zum Beispiel das automatische Ausführen von Windows-Updates, deaktiviert, um eine CPU-Auslastung von nahezu 0% zu erreichen.

Implementierung der Datenmodelle: Innerhalb dieser Datenbankumgebung wurden dann drei Datenbanken mit jeweils unterschiedlichen Datenmodell generiert. Einerseits eine spaltenorientierte Datenbank, andererseits eine zeilenorientierte Datenbank, beide auf dem Starschema basierend, und letztlich noch eine zeilenorientierte Datenbank, basierend auf dem Data Vault Schema. Den grundlegenden Aufbau der einzelnen Datenbanken kann man den Abbildungen 2 und 5 entnehmen.

Füllen der Datenbanken: Die Datenbanken wurden dann mit den gleichen Datensätzen gefüllt, sodass die Faktentabelle im Starschema 30000 Datensätze beinhaltete. Die Daten wurden größtenteils automatisch über die Website www.generatedata.com generiert, da diese über Templates für numerische Werte, wie Ids und Preise, aber auch über textarische wie Namen, Berufsgruppen und Städte, verfügte.

5.2 Erstellung des Abfrageplans

Nachdem die Testumgebung aufgebaut wurde, muss ein Abfrageplan definiert werden, dessen Umsetzung letztendlich auf beide Datenmodelle angewendet wird. Dieser Abfrageplan besteht aus zwei Schritten:

Definition des Leistungsindikators: Innerhalb dieses Tests wird in erster Linie die Performance mit Hilfe des Leistungsindikators Laufzeit gemessen und bewertet. Die Laufzeit ist ein Zeitmaß, welches die verstrichene Zeit vom Abschicken eines SQL-Befehls bis zur finalen Rückmeldung des Servers über eine erfolgreiche Berechnung des Befehls misst. Die Zeiteinheit beträgt dabei Millisekunden. Da sich die Laufzeiten bei vielfacher Anwendung eines SQL-Befehls hintereinander einpegeln, ist es möglich, dass es zum sogenannten Cache-Phänomen kommt. Caching wird verwendet, um Daten, die häufig vom Anwender nachgefragt werden, beim Client zwischenzuspeichern, damit die Laufzeit verringert wird. Um dieses Phänomen zu minimieren, wird eine „Cold-Start"-Messung durchgeführt, d.h., dass nach dem Start des Servers bei jeweils erster Anwendung der Abfrage der erste Wert als Referenz für die Laufzeit genommen wird. [Klaßen et al. 2004]

Definition der Abfragen: Um erste Messungen bewerten zu können, werden verschiedene Arten von Abfragen auf den Datenbanken ausgeführt. Dabei handelt es sich zum einen um OLTP-Abfragen. Diesbezüglich um die Punktabfrage, welche nach allen Objekten auf einen bestimmten Wert hin sucht, und die Bereichsabfrage, da diese eine Suche nach allen Objekten bezüglich eines bestimmten Bereichs durchführt. Zum anderen werden OLAP-Operationen angewendet, um, bezogen auf typische Anfragemuster von Data Warehouse Anwendungen, auf Aggregatfunktionen, unterschiedliche Granularitäten, und auch der Berechnung von Teil- und Gesamtsummen einzugehen. Typische OLAP-Operationen können in vier Klassen unterteilt werden:

- Pivot/Rotate
- Drill Down/Roll Up
- Drill Across
- Slice/Dice

Da ein SQL-DBMS als Backend benutzt wird, müssen diese OLAP-Abfragen abhängig von der Art des Realisierungsschemas in SQL-Abfragen umgewandelt werden. Für das Starschema wird daher das Muster des Star-Joins herangezogen. Bei einem Star-Join muss die Faktentabelle einen „sternförmigen" Verbund mit den Dimensionstabellen bilden, auf denen dann Restriktionen formuliert werden, sowie Aggregationen über Kennzahlen in Verbindung mit Gruppierungen. Ausformuliert bedeutet es, dass eine typische Data Warehouse Abfrage aus einer Select-Klausel mit

Aggregation, einer From-Klausel mit Fakten-, Dimensionstabellen und Joins, einer Where-Klausel mit Restriktionen und einer Gruppierungs-Klausel besteht. Auf Grund von bewiesenen Nachteilen beim Nutzen des einfachen Group By-Operators, wurden seit SQL:2003 Erweiterungen für Gruppierungsfunktionen aufgenommen. Im weiteren Verlauf wird vor allem der Cube-Operator benutzt, da dieser Teil- und Gesamtsummen berechnet, so dass aus einer gegebenen Menge von Gruppierungsattributen alle möglichen Gruppierungskombinationen generiert werden. [Köppen et al. 2014]
Im Folgenden werden die verschiedenen Abfragetypen einzeln durch jeweils eine Abfrage auf die drei Datenbanken ausgeführt und anhand der Laufzeit dargestellt.

1. Punktabfrage: Es wird eine Punktabfrage umgesetzt, bei der alle Daten aus der Tabelle Verkauf aufgelistet werden sollen, bei denen der Umsatz 99.99€ beträgt.
Die Abfrage sieht beim Starschema wie folgt aus:

*SELECT * FROM Verkauf WHERE V_Umsatz=99.99;*

Beim DataVault ist dafür folgende Abfrage nötig:

SELECT KundenHashKey, ZeitHashKey, ProduktHashKey, RegionHashKey, Umsatz FROM LinkVerkauf, SatelliteVerkauf
WHERE LinkVerkauf.VerkaufHashKey = SatelliteVerkauf.VerkaufHashKey
AND SatelliteLink.Umsatz = 99.99;

Um die gleiche Anzahl an Attributen wie bei der Star-Query zu bekommen, müssen bei der Data Vault Umsetzung in der Select-Klausel all die Attribute ausgewählt werden, die im Starschema in der Faktentabelle stehen, da die Tabelle LinkVerkauf bzw. SatelliteVerkauf im DataVault über mehr Spalten verfügt als die Faktentabelle Verkauf im Starschema. Während beim Starschema die einzelnen Business Keys und der Umsatz ausgegeben werden, werden im Data Vault die aus den Business Keys generierten Hash-Schlüssel in die Ergebnismenge aufgenommen. Um die Business Keys im Data Vault zu ermitteln, müssten weitere Joins über die Hubs laufen, was hier den Sinn der Punktabfrage verfehlen würde.

2. Bereichsabfrage: Mit Hilfe der Bereichsabfrage soll eine Liste mit allen Umsätzen und den dazugehörigen Produktnamen generiert werden, welche innerhalb eines Monats zwischen dem 25.11.2015 und dem 25.12.2015 erzielt wurden.
Die Abfrage im Starschema sieht wie folgt aus:

```
SELECT V_Umsatz, P_Name FROM Verkauf, Zeit, Produkt
WHERE V_Zeit_ID=Z_ID
V_Produkt_ID = ProduktKey
and Z_Datum between '2015-11-25' and '2015-12-25';
```

Beim DataVault ist dafür folgende Abfrage nötig:

```
SELECT Umsatz, P_Name
FROM SatelliteVerkauf, LinkVerkauf, HubZeit, Hub Produkt, Satellite Produkt
WHERE SatelliteVerkauf.VerkaufHashKey= LinkVerkauf.VerkaufHashKey
and LinkVerkauf.ZeitHashKey=HubZeit.ZeitHashKey
and LinkVerkauf.ProduktHashKey=HubProdukt.ProduktHashKey
and HubProdukt.ProduktHashKey = SatelliteProdukt.ProduktHashKey
and HubZeit.ZeitID between '20151125' and '20151225';
```

Wenn man davon ausgeht, dass man den Business Key der Zeit, also die ZeitID, kennt und weiß, dass dieser letztendlich auch als Datum gelesen werden kann, sieht die Abfrage wie oben angegeben aus.

3. OLAP-Abfrage: Mit Hilfe dieser Abfrage sollen sowohl der Name des Produkts, die Summe des Umsatzes, den jedes Produkt erzielt hat, als auch der Name der Kunden, der diese Produkte gekauft haben, aufgelistet werden, um zu sehen, ob ein Kunde ein Produkt mehrmals gekauft. Dabei soll es sich nur um Produkte handeln, die zur Produktgruppe „Kleidung“ gehören. Weiterhin wird diese Auflistung nach den Attributen des Kundennamens und des Produktnamens gruppiert, indem der CUBE-Operator verwendet wird.
Die Abfrage an das Starschema sieht wie folgt aus:

```
SELECT Kunde.K_Name, Produkt.P_Name, sum(Verkauf.Umsatz)
FROM Verkauf, Kunde, Produkt
WHERE Kunde.K_ID = Verkauf.V_Kunden_ID
```

```
And Verkauf.V_Produkt_ID = Produkt.P_ID
and Produkt.P_Gruppe='Kleidung'
GROUP BY CUBE (Kunde.K_Name,Produkt.P_Name);
```

Beim DataVault ist dafür folgende Abfrage nötig:

```
SELECT SatelliteKunde.Kunden_Name, SatelliteProdukt.Produkt_Name,
sum(SatelliteVerkauf.Umsatz)
FROM LinkVerkauf, SatelliteVerkauf, SatelliteProdukt, SatelliteKunde
WHERE SatelliteKunde.KundenHashKey = LinkVerkauf.KundenHashKey
and SatelliteVerkauf.VerkaufHashKey = LinkVerkauf.VerkaufHashKey
and LinkVerkauf.ProduktHashKey = SatelliteProdukt.ProduktHashKey
and SatelliteProdukt.P_Gruppe='Kleidung'
GROUP BY CUBE (SatelliteKunde.Kunden_Name, SatelliteProdukt.Produkt_Name);
```

Bei der Erstbetrachtung der Abfragen fällt auf, dass die für das Data Vault gestellten Abfragen länger scheinen. Dies äußert sich zum einen darin, dass die Attributnamen im Data Vault anders gewählt wurden als im Starschema. Jedoch kann man auch bezüglich der Anzahl der Verknüpfungen bereits erste Unterschiede erkennen. Die Abfragen beim Data Vault haben teilweise mehr Verknüpfungen als die beim Starschema. Während im Starschema alle Attribute eines Objekts innerhalb einer Tabelle verfügbar sind, werden für die Agilität beim Data Vault mehrere Satelliten mit beschreibenden Attributen für ein Objekt akzeptiert. Um diese Situation zu verdeutlichen, wird die Entität „Kunde“ um die Attribute „IBAN“ und „BIC“ und „Name der Bank“ erweitert. Im Starschema würde lediglich die Tabelle „Kunde“ um ein Attribut erweitert, wobei im Data Vault ein neuer Satellit „SatelliteKundenKonto“ entstehen könnte, welcher die beschreibenden Elemente des Attributs beinhaltet. Dies hat einerseits den Vorteil, dass vorhandene Satelliten nicht überarbeitet und aktualisiert werden müssen (unter anderem auch die Prozesse zum Laden neuer Daten in der Ladephase), andererseits kann es zum Nachteil durch eine größere Anzahl an Joins bei bestimmten Abfragen und somit zu Performance-Einbußen führen. Nimmt man an, dass bei der OLAP-Abfrage von oben nicht nur Kundenname, Produktname und der aggregierte Verkaufspreis ausgegeben werden sollen, sondern auch die IBAN eines jeden Kunden aus der neu generierten Tabelle „SatelliteKundekonto“, sieht die Abfrage für das Starschema wie folgt aus:

*SELECT Kunde.K_Name, **Kunde.K_IBAN**, Produkt.P_Name, sum(Verkauf.Umsatz)*
FROM Verkauf, Kunde, Produkt
WHERE Kunde.K_ID = Verkauf.V_Kunden_ID
and Verkauf.V_Produkt_ID = Produkt.P_ID
and Produkt.P_Gruppe='Kleidung'
*GROUP BY CUBE (Kunde.K_Name, Produkt.P_Name, **Kunde.K_Kundenkonto**);*

Für das DataVault würde die Abfrage folgendermaßen beschrieben werden:

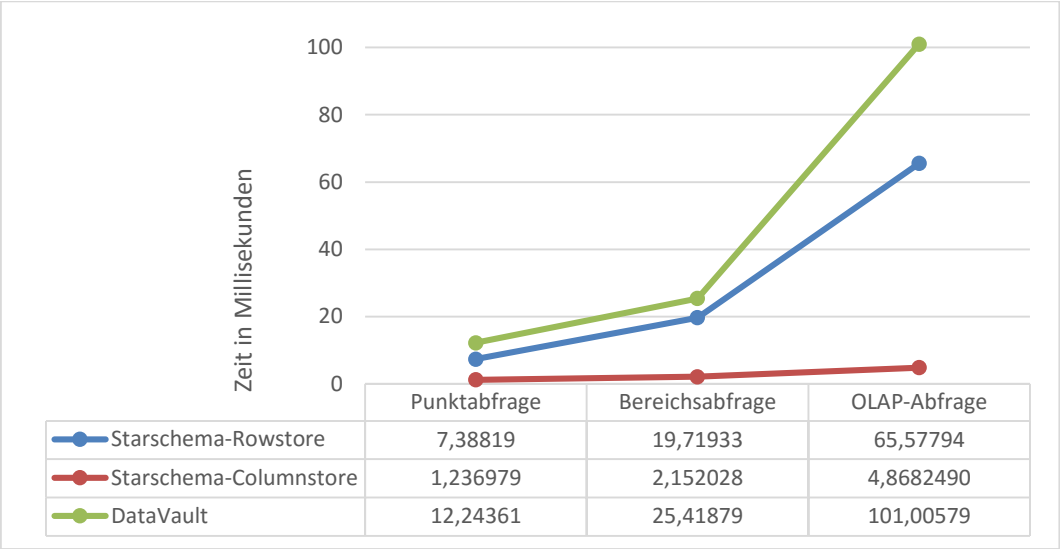
*SELECT SatelliteKunde.Kunden_Name, **SatelliteKundenKonto.IBAN**, SatelliteProdukt.Produkt_Name, sum(SatelliteVerkauf.Umsatz)*
*FROM SatelliteVerkauf, LinkVerkauf, SatelliteKunde, **SatelliteKundenKonto,** SatelliteProdukt*
WHERE SatelliteKunde.KundenHashKey = LinkVerkauf.KundenHashKey
and SatelliteKundenKonto.KundenHashKey = LinkVerkauf.KundenHashKey
and LinkVerkauf.ProduktHashKey = SatelliteProdukt.ProduktHashKey
and SatelliteVerkauf.VerkaufHashKey = LinkVerkauf.VerkaufHashKey
and SatelliteProdukt.P_Gruppe='Kleidung'
*GROUP BY CUBE (SatelliteKunde.Kunden_Name, SatelliteProdukt.Produkt_Name, **SatelliteKundenKonto.IBAN**);*

An diesem Beispiel kann man deutlich erkennen, dass bei komplexeren Abfragen über mehrere beschreibende Elemente aus verschiedenen Satelliten wesentlich mehr Verknüpfungen bei einer Data Vault Abfrage notwendig sind als beim Starschema.

5.3 Performance-Messung

Wie in Abschnitt 5.2 beschrieben, wird zur ersten Beurteilung der Performance die Laufzeit der verschiedenen Abfragen gemessen. Dafür wurden die eben definierten Abfragen in der Testumgebung auf die verschiedenen Datenbanken angewendet. Alle Abfragen werden einzeln nach dem „Cold-Start“-Prinzip ausgeführt, so dass Caching-Effekte minimiert werden. Abbildung 9 illustriert das Ergebnis dieser ersten Laufzeitmessung. Während das Data Vault die niedrigste Performance, d.h. die höchste Laufzeit, aufweist, hat die Variante des Column Stores die niedrigste Laufzeit und ist somit bezogen auf die Testumgebung das performanteste Modell. Weiter-

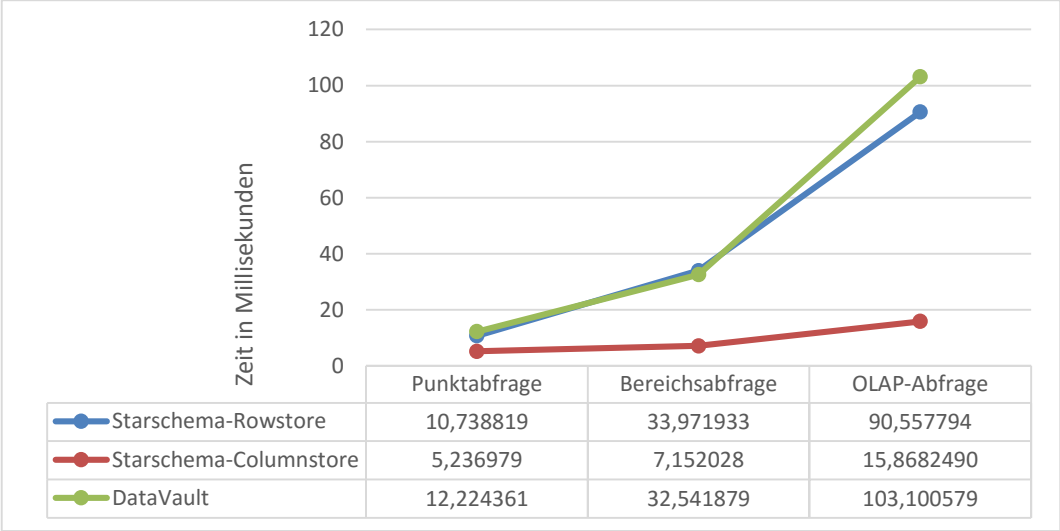
hin ist zu erkennen, dass OLAP-Abfragen deutlich komplexer sind und dementsprechend eine größere Laufzeit haben.



	Punktabfrage	Bereichsabfrage	OLAP-Abfrage
Starschema-Rowstore	7,38819	19,71933	65,57794
Starschema-Columnstore	1,236979	2,152028	4,8682490
DataVault	12,24361	25,41879	101,00579

Abbildung 9: Ergebnis der ersten Laufzeitmessung

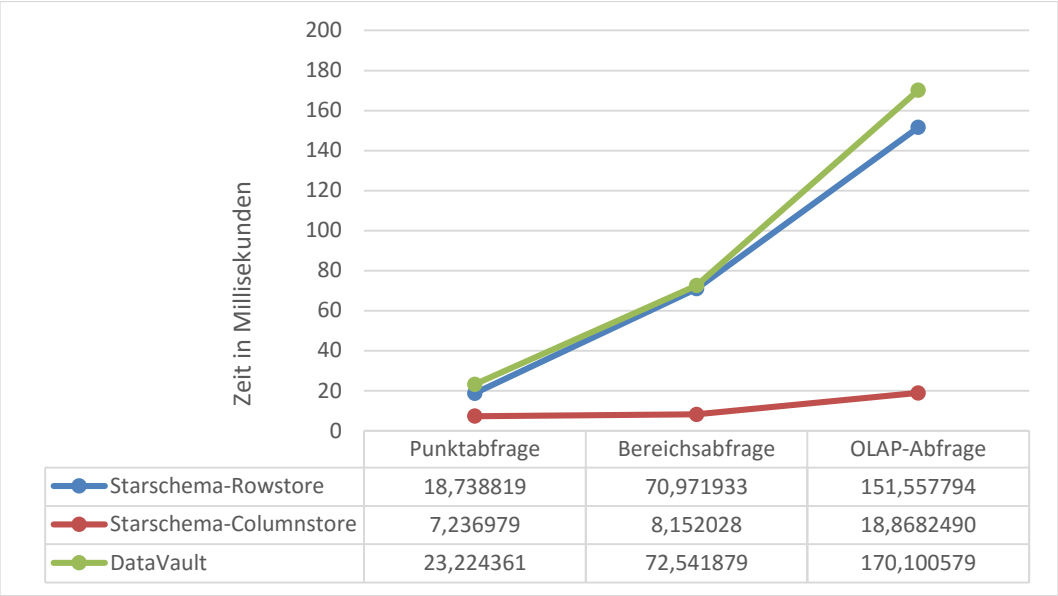
Um diese ersten Erkenntnisse zu festigen wurden zwei weitere Messungen durchgeführt. Dabei wurde vor jeder Messung der Datenbestand in der Faktentabelle um je 50000 Datensätze erhöht, sodass auch auf verschieden großen Datenbeständen eine Analyse erfolgte.



	Punktabfrage	Bereichsabfrage	OLAP-Abfrage
Starschema-Rowstore	10,738819	33,971933	90,557794
Starschema-Columnstore	5,236979	7,152028	15,8682490
DataVault	12,224361	32,541879	103,100579

Abbildung 10: Ergebnis der zweiten Laufzeitmessung

Abbildung 10 illustriert den zweiten Testdurchlauf. Hier bestätigen sich die Ergebnisse der ersten Messung. Die Column Stores haben die beste Performance während das Data Vault hauchdünn das Nachsehen gegenüber dem Starschema hat. In einer dritten Messung wurde die Menge an Datensätzen in der Faktentabelle abermals um 50000 Datensätze erhöht. Auch hier ist der gleiche Trend wie zuvor zu erkennen. Abbildung 11 zeigt die Ergebnisse des dritten Testdurchlaufs.



	Punktabfrage	Bereichsabfrage	OLAP-Abfrage
Starschema-Rowstore	18,738819	70,971933	151,557794
Starschema-Columnstore	7,236979	8,152028	18,8682490
DataVault	23,224361	72,541879	170,100579

Abbildung 11: Ergebnis der dritten Laufzeitmessung

5.4 Bewertung der Ergebnisse

Der Grund für die Ausführung einer ersten übersichtlich gehaltenen qualitativen Messung bestand in dem Erlangen erster Erkenntnisse über das Verhalten der Abfragebearbeitung des Data Vault Modells gegenüber dem Starschema und den Column Stores auf Basis messbarer Werte. Die Column Stores scheinen im Vergleich zu den anderen beiden Varianten wesentlich performanter. Das liegt einerseits an den Komprimierungsverfahren, andererseits aber auch an den vom SQL Server stetig weiterentwickelten Column Store Index, welcher die Abfragebearbeitung effizienter gestaltet. Da die Column Stores bekanntlich Schwächen beim Arbeiten auf vielen Spalten haben, wie beispielsweise beim Einfügen und Schreiben vieler Tupel, ist ein Vergleich der Performance von Aktualisierungsprozessen im weiteren Verlauf zu bedenken. Weiterhin ergibt sich durch diesen ersten quantitativen Vergleich die Annahme, dass trotz oder gerade weil das Data Vault Modell minimal schlechtere Laufzeiten bei der Abfragebearbeitung auf einen sehr überschaubaren Datenbestand hatte, es auch bei größeren Datenbeständen zu schlechteren Performance-Ergebnissen gegenüber dem Starschema führt.

6. Qualitativer Datenmodell-Vergleich

Ein erster quantitativer Vergleich anhand der Performance zwischen dem Star- und dem Data Vault Schema hat gezeigt, dass das Starschema eine minimal bessere Performance aufweist. Der qualitative Vergleich wurde anhand des aus der Literatur gewonnenen Kriterienkatalogs und den bei der Erstellung und Ausführung des quantitativen Vergleichs gewonnenen Erkenntnissen gezogen. In Tabelle 1 wird der Vergleich der in Abschnitt 4 definierten qualitativen Kriterien zusammengefasst.

Tabelle 1:Qualitativer Vergleich Starschema - Data Vault

Qualitative Kriterien	Starschema	Data Vault Schema
Multidimensionale Sicht	Ja	Nein
Erweiterbarkeit	Updates auf existierenden Tabellen	Flexibel; Einfügen neuer Komponenten möglich, ohne bestehende Tabellen anzufassen
Uneingeschränkte Operationen	Ja	Nicht im Raw Vault, aber im Business Vault
Abfragekomplexität	Einfach, wenige Joins	Schwer, vergleichsweise viele Joins
Struktur des Datenmodells	Klein, wenige Tabellen	Bei vielen Business Objekten schnell unübersichtlich

Multidimensionale Sicht: Das Starschema basiert auf dem dimensionalen Datenmodell aus Abschnitt 2.1. Es besteht demnach aus einer Faktentabelle mit metrischen Werten und einer Anzahl an Dimensionen. Bei Betracht des Data Vault Schemas in Abbildung 5 fällt auf, dass es dem Starschema aus Abbildung 2 sehr ähnlich ist. Im Data Vault Modell gibt es einen Link, der mit allen Hubs durch Fremdschlüssel verknüpft ist; im Starschema eine Faktentabelle, die gleichermaßen mit allen Dimensionen in Beziehung steht. Während die beschreibenden Attribute im Starschema jedoch in den Dimensionstabellen verankert sind, werden beim Data Vault Modell Satelliten hinzugefügt. Ein weiterer großer Unterschied besteht letztendlich darin, dass das originale Data Vault Modell im Raw Vault keine aggregierten Kennzahlen, wie in diesem Beispiel es der Umsatz ist, enthält. Aufgrund dessen ist die multidimensionale Sicht im Data Vault nicht gegeben. Bei späteren Data Vault Modellen (z.B. Abbildung 17) wird auch deutlicher, wie weit dieses Modell von der dimensionalen Sicht weg ist. Geht man allerdings nicht vom Data Vault Mo-

dell an sich, sondern von der ganzen Data Vault Architektur aus, unterstützt es schließlich doch die dimensionale Sicht in dem Information Mart Layer, in welchem die Data Marts als Starschemas abgespeichert werden können. [Linstedt et al. 2015]

Erweiterbarkeit: Das Data Vault Modell mit seinen drei Komponenten geht mit hoher Flexibilität bei Veränderungen einher. Satelliten, Hubs und Links können beliebig dem Datenmodell hinzugefügt werden, ohne dabei die vorhandenen Tabellen mitsamt den Attributen adaptieren zu müssen, während beim dimensionalen Modell Veränderungen teilweise unabdinglich sind. Nimmt man zum Beispiel an, dass zu einem bestimmten Zeitpunkt ein Kunde umzieht und sich somit die Adresse und Telefonnummer ändert, dann müsste man im Starschema je nachdem welchem Typ die Dimension entspricht, Änderungen vornehmen. Ist es eine Typ-1 Dimension legt man einen komplett neuen Kunden samt Primärschlüssel an, was zu falschen Analysen führen kann. Bei einer Typ-2 Dimension würde man neue Attribute für die gesamte Tabelle mit neuer Adresse und neuer Telefonnummer einfügen und die zu aktualisierenden Daten unter der alten Adresse bzw. unter der alten Telefonnummer abspeichern. Bei einer Typ-3 Dimension würde man die aktuelle Adresse und Telefonnummer überschreiben, wodurch allerdings dann die Historisierung der Daten leidet, da mit den alten Kundendaten keine Analysen mehr möglich sind. Die Typ-4 Dimension würde eine neue Tabelle einfügen, die die historischen Daten beinhaltet, während die Dimensionstabelle nur die aktuellsten Daten speichert. Ganz egal welcher Ansatz infrage kommt, würde die Tabelle somit angefasst und überarbeitet werden. Im Data Vault Modell wäre ein Ansatz, einen neuen Satelliten zum Hub Kunden hinzuzufügen, der die Adress- und Telefonänderungen bearbeitet. Somit könnte man auch einfache Vergleiche zwischen dem Verhalten des Kunden vor und dem Verhalten nach dem Umzug erzielen. Diese Flexibilität beim Datenmodell kann allerdings auch als Nachteil gesehen werden. Das Data Vault Modell wächst sehr rasant an und kann schnell unübersichtlich werden. Um auf Daten von mehreren Satelliten zuzugreifen, ist eine Vielzahl an Joins beziehungsweise kartesischen Produkten nötig, was im vorigen Abschnitt schon belegt wurde. [Linstedt et al. 2015]

Uneingeschränkte Operationen: Kreuzdimensionale Operationen wie CUBE oder ROLLUP sind vom System automatisch durchgeführte Berechnungen, die durch die Navigation durch die einzelnen Hierarchiestufen der Dimensionen nötig werden. Wie die Definition schon zeigt, sind diese eigentlich systemabhängig und werden nicht direkt vom Datenmodell eingeschränkt. Nichtsdestotrotz kann man feststellen, dass ein Data Vault Modell im Raw Vault nicht für solche kreuzdimensionalen Operationen ausgelegt ist, während man im Business Vault sich solcher Operationen bedienen kann, um Vorberechnungen durchzuführen. Das Starschema sollte in

einer solchen Korrektheit implementiert sein, dass auf diesem uneingeschränkt alle kreuzdimensionalen Operationen durchgeführt werden können. [Kimball et al. 2011; Linstedt et al. 2015]

Abfragekomplexität: Die in Abschnitt 5.2 formulierten Abfragen geben erste Anhaltspunkte dafür, dass die Abfragen auf eine im Data Vault Schema implementierte Datenbank komplizierter sind. Auch wenn der Grundaufbau gleichbleibt, kann bei der Erweiterung des Schemas die Länge von Abfragen mitwachsen. Durch die separate Handhabung der Business Keys und beschreibenden Attribute eines Objekts sind mehr Verknüpfungen vonnöten, um an gewisse Daten zu gelangen. Durch die hohe Erweiterbarkeit und Flexibilität des Schemas können diese Verknüpfungen die Abfragen enorm in die Länge ziehen und einem Anwender schnell unübersichtlich erscheinen lassen. Die Anwender müssten die komplexen Data Vault Modelle sehr gut kennen, um Abfragen auf das gesamte Modell anzuwenden. Anders verhält es sich beim Starschema. Hier wächst die Zahl der Joins linear mit der Anzahl an Dimensionen. Um Restriktionen auf einzelne Dimensionen anzuwenden oder Aggregationen von bestimmten Attributen einer Dimension vorzunehmen, müssen lediglich die Dimensionstabellen, in denen sich diese Attribute befinden, bekannt sein und eventuell über die Faktentabelle verknüpft werden. Als Beispiel kann hier die OLAP-Abfrage aus Abschnitt 5.2 herangezogen werden. Während beim Starschema lediglich zwei Dimensionstabellen und eine Faktentabelle verknüpft werden müssen, um den Kundennamen, Produktnamen und den summierten Umsatz der vom Kunden gekauften Produkte zu ermitteln, müssen beim Data Vault Modell im gleichen Atemzug sechs Tabellen miteinander verknüpft werden. Die Auswirkungen dessen wurde bereits im quantitativen Vergleich in Abschnitt 5.4 anhand der Laufzeitmessungen dargestellt. [Linstedt et al. 2015]

Struktur des Datenmodells: Die Struktur des Datenmodells korrespondiert mit den Kriterien der Erweiterbarkeit und Verständlichkeit der Abfragen. Das Starschema ist relativ klein gehalten und dank einer im Mittelpunkt stehenden Faktentabelle, die mit jeder Dimension in Beziehung steht, gut strukturiert und übersichtlich. Dies hat natürlich zur Folge, dass, wie oben beschrieben, es in seiner Flexibilität einbüßt, dafür aber für den Anwender in Bezug auf die Ableitung von Abfragen verständlicher ist. Durch die großzügige Erweiterbarkeit, aber auch durch die unterschiedliche Handhabung von den einzelnen Komponenten, hier sind im Speziellen die unterschiedlichen Satellit-Typen zu nennen, können Datenmodelle in deren Struktur extrem variieren und im Gegensatz zum Starschema recht schnell unübersichtlich werden. [Linstedt et al. 2015]

Auf Grundlage des qualitativen und quantitativen Vergleichs schneidet das Starschema wesentlich besser ab. Es kann sowohl in der Performance, als auch der Verständlichkeit und Übersichtlichkeit von Abfragen und der Struktur des Datenmodells überzeugen. Daher ist anzunehmen,

dass das Data Warehouse, implementiert auf Basis des Starschemas, aus Sicht der Endnutzer eine größere Anerkennung findet. Andererseits überzeugt das Data Vault Modell mit hoher Flexibilität bei Erweiterungen, was den ganzen Ladeprozess von den Datenquellen über die Staging Area bis hin zum Data Vault vereinfachen kann. Aus Sicht der Entwickler kann dadurch die Instandhaltung des ganzen Data Warehouses optimiert werden. [Köppen et al. 2014; Linstedt et al. 2015]

Der Vergleich liefert zwar erste Erkenntnisse über die Vor- und Nachteile beider Modelle, ist jedoch nicht ausreichend, um die Ergebnisse zu manifestieren. Das liegt einerseits daran, dass ein Teil der qualitativen Anforderungen aus „weichen“ Anforderungen besteht und somit mit der Subjektivität des Autors bewertet wurden, andererseits aber auch daran, dass die Datenbanken relativ klein gehalten sind und die Anfragen auf diesen nicht alle Bereiche funktional und selektiv abdecken. Um der Willkür des Autors und der Frage nach eventuell absichtlich herbeigeführten Ergebnissen entgegenzuwirken, wird daher ein Benchmark herangezogen, der für Analysen von Datenbanksystemen bekannt und akzeptiert ist. Mit dessen Ausführung sollen die im quantitativen Vergleich gewonnenen Erkenntnisse entweder manifestiert oder widerlegt werden.

7. TPC-H Benchmark

Als Benchmarks werden im allgemeinen Programme bezeichnet, welche zur Analyse von Ergebnissen und Prozessen mittels vordefinierten Bezugswert eingesetzt werden. Sie sind weit verbreitet und in den unterschiedlichsten Gebieten, zum Beispiel in der Betriebswirtschaft, der Technologie-Branche, oder auch der IT-Branche, anwendbar.

Computer-Benchmarks beschreiben Programme, mit deren Hilfe man Computersysteme auf ihre Leistung hin überprüfen kann. Die Sparte der Computer-Benchmarks kann weiter klassifiziert werden. Im Hinblick auf den Vergleich der Datenmodelle sind die Datenbank-Benchmarks von Bedeutung, insbesondere der TPC-H Benchmark, der von dem Transaction Processing Performance Council entwickelt wurde.

Das Transaction Processing Performance Council (TPC) ist eine Organisation, bestehend aus Führungskräften vieler Unternehmen der IT-Branche, die dem Zweck dient, Benchmarks für Transaktionsprozesse zu schreiben, um anhand derer objektive und verifizierte Methoden zu Performanceanalysen innerhalb der Datenbankwelt zu verbreiten. Abhängig vom Anwendungsszenario bietet das TPC aktuell 10 verschiedene Benchmarks an (Stand: 05.03.16), die zur Analyse von klassischen OLTP-Systemen (TPC-C, TPC-E), über Entscheidungssysteme (TPC-H, TPC-DS, TPC-DI), bis hin zur Virtualisierung (TPC-VMS, TPCx-V) reichen.

Der TPC-H Benchmark wiederum ist ein Benchmark zum Analysieren entscheidungsunterstützender Systeme auf OLTP-Basis, der jedoch durch komplexe anwendungsbezogene OLAP-Abfragen auf sehr großen Datenmengen besticht und durch die Metrik „Abfragen pro Stunde bei einer bestimmten Datenbankgröße (QphH@Size)“ zur Analyse von Data Warehouse Systemen herangezogen werden kann. Dabei simuliert der Benchmark ein Handelsunternehmen, bestehend aus Kunden, Standorten und Zulieferern. Insgesamt spezifiziert der Benchmark 8 Tabellen in der 3NF. (Siehe Abbildung 12)

Zur Generierung der Datenmengen wird das Programm DBGEN vom TPC bereitgestellt, wobei mit Hilfe eines Skalierungsfaktors verschiedene Datenbankgrößen vordefiniert und erstellt werden können. 6 der 8 Tabellen wachsen linear mit dem Skalierungsfaktor mit. Die anderen Zwei weisen stetig eine konstante Größe auf. Ebenso wie bei der Generierung der Daten, können auch die komplexen Abfragen mit Hilfe eines Programms (QGEN) automatisch generiert werden, welche schließlich den komplexen OLAP-Workload simulieren. [TPC 2014]

Zum Testumfang und -ablauf gehören 22 langlaufende und komplexe Abfrage-Templates (siehe Anhang A) sowie 2 Aktualisierungs-Prozesse (insert, delete), die einerseits im Power-Test als eine Anfragesequenz hintereinandergeschaltet ausgeführt werden und andererseits im Through-

put-Test parallel ausgeführt werden, um die Nebenläufigkeit des Systems zu testen. Auch hier gilt, je größer die Datenbank, desto mehr nebenläufige Prozesse müssen stattfinden.

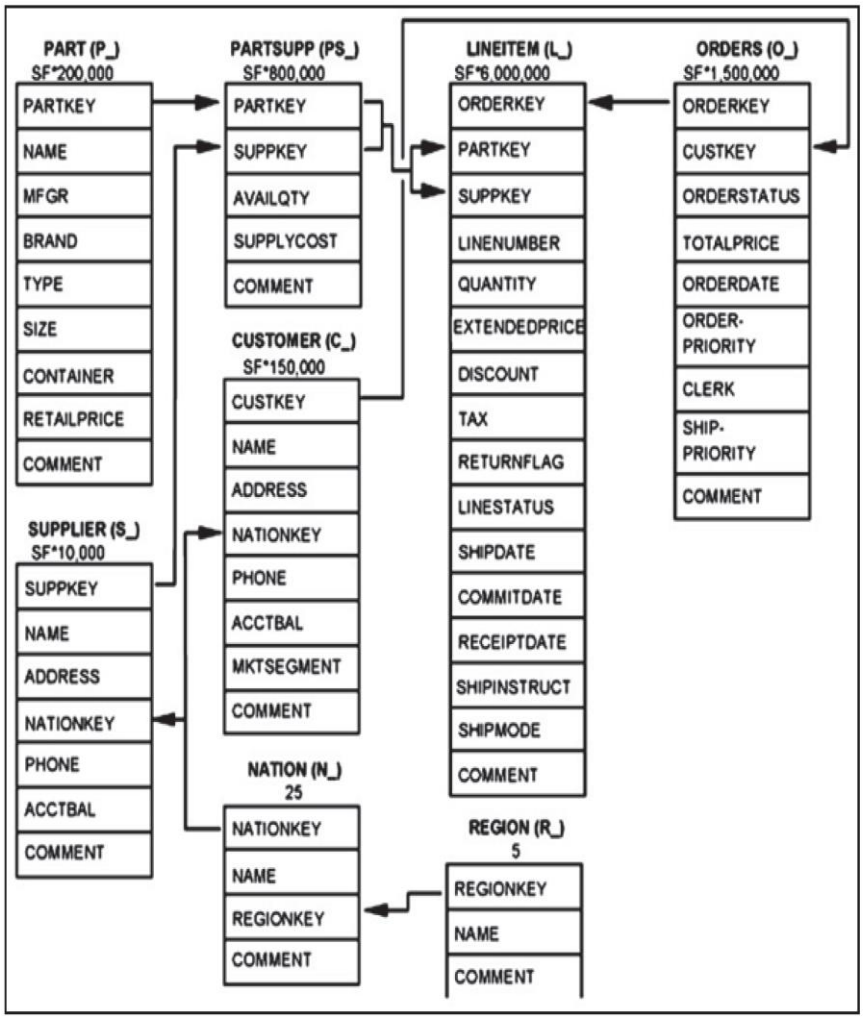


Abbildung 12:TPC-H Schema

Während also die 22 Abfragen in ihrer Struktur vordefiniert sind, können die zwei Aktualisierungsprozesse relativ flexibel umgesetzt werden, auch wenn es da gewisse Vorgaben gibt.
Um die Testergebnisse veröffentlichen zu dürfen, muss ein sogenannter „Full Disclosure Report“ erstellt werden. Neben den gewonnenen Testergebnissen gehören in diesen Bericht unter anderem auch die Skripte und Quelldateien zur Reproduktion der Datenbanken, eine Zusammenfassung, die Einstellungen und Parameter der Software und die Charakteristika der Hardware. [TPC 2014]

8. TPC-H basierter quantitativer Vergleich

Der TPC-H Benchmark kann aufgrund seiner OLTP-Basis nicht ohne Anpassungen zum Vergleich der beiden Datenmodelle des Star- und Data Vault Schemas eingesetzt werden. Das fiktive Handelsunternehmen muss erst in Form eines Starschemas beziehungsweise Data Vaults umgesetzt werden, damit die Durchführung des quantitativen Vergleichs zwischen beiden Modellen möglich ist.

In Kapitel 8.1 wird zunächst die Umsetzung für das Starschema beschrieben, welche bereits von [O'Neil et al. 2009] entwickelt wurde und in der Datenbankliteratur zu finden ist. Kapitel 8.2 beschreibt folglich das auf dem Data Vault basierende Modell. Anschließend werden in Kapitel 8.3 die Abfragen, welche zur Performance-Messung herangezogen werden, jeweils für beide Benchmarks beschrieben, bevor es in den Abschnitten 8.4-8.8 zur Anwendung und Auswertung der eigentlichen Messung kommt.

8.1 TPC-H basiertes Starschema

Eine fehlerfreie Anwendung der Abfragen des TPC-H Benchmarks auf das Starschema ist schon dadurch nicht gegeben, da alle Tabellen im Benchmark der 3NF entsprechen, auf welche bei der Umsetzung im Starschema bewusst verzichtet wird. [O'Neil et al. 2009] haben sich der Problematik angenommen und einen Starschema-Benchmark (SSB) entwickelt, welcher auf dem TPC-H Benchmark basiert. Abbildung 13 illustriert die Umsetzung des Star-Schema Benchmarks.

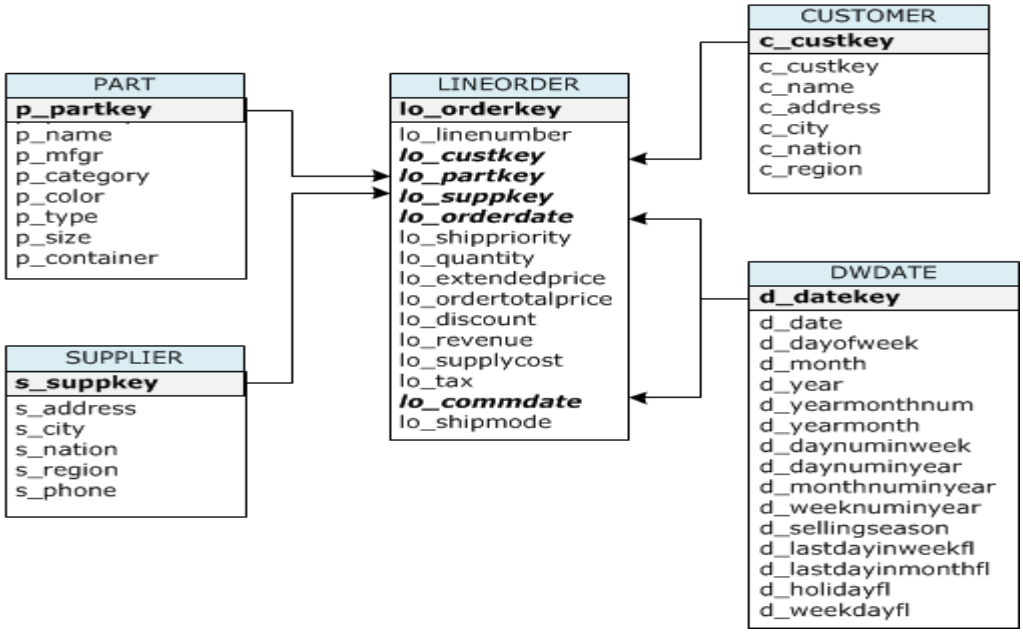


Abbildung 13: Starschema basierend auf dem TPC-H Modell

Im Vergleich zum TPC-H Benchmark fällt sofort auf, dass die Tabellen „lineitem" und „orders" zusammengesetzt die Faktentabelle „lineorder" definieren. Dabei wurden verschiedene Attribute

(u.a. O_Orderstatus, L_Receiptdate, L_Shipdate, etc.) gelöscht, die eine Verzögerung zum Einfügen neuer Tupel verhindern würden. In anderen Worten, würde man den Status der Bestellung oder auch das Versanddatum in die Faktentabelle aufnehmen, dann müsste man mit dem Einfügen aller Bestellungsattribute in die Faktentabelle warten, bis diese Events aufgetreten sind, was zur Folge hätte, dass man Abfragen über die Existenz einer solchen Bestellung erst verzögert ausführen könnte. Daher ist es sinnvoll Attribute, die solch eine Verzögerung in Kauf nehmen, nicht in das Konstrukt mit einzubeziehen. Des Weiteren werden alle Textfelder mit Freizeichen (L_Comment, O_Comment, L_Shipinstruct) gelöscht, da diese von typischen Data Warehouse Abfragen nicht geparst und aggregiert werden können. Außerdem wird ein neues aggregiertes Attribut „Revenue“ hinzugefügt, welches sich aus der Multiplikation von „ExtendedPrice“ und „Discount“ zusammensetzt.

Die Anzahl an Dimensionen wird durch die Tabelle „DWDate“ erweitert, da so gut wie jedes Data Warehouse zeitbezogen ist. [Kimball et al. 2011]

Die im TPC-H vorhandene Tabelle PartSupp wird komplett gelöscht, da diese eine periodische Granularität aufweist, während die Tabellen „lineitem“ und „order“ die feinste zeitliche Granularität aufweisen. Eine separate Faktentabelle ohne direkte Verknüpfung zur Faktentabelle „LineOrder“ ist daher vorteilhafter.

Weiterhin werden innerhalb jeder Tabelle die Datentypen einiger Attribute angepasst, Attribute hinzugefügt oder auch Attribute gelöscht. Dabei beziehen sich O’Neil et al. ganz auf die dimensionale Umsetzung von Kimball. Die einzelnen Datentyp-Änderungen sind [O’Neil et al. 2009] zu entnehmen.

8.2 TPC-H basiertes Data Vault Modell

Das Data Vault Schema kann auf einfache Weise aus dem TPC-H Modell generiert werden, da im Gegensatz zum Starschema weder auf verschiedene Granularitätsebenen, noch auf die Normalisierung Bezug genommen werden muss. Die Beachtung derer erfolgt nicht auf Data Vault-Ebene, sondern erst im Information Layer beim Erstellen der einzelnen dimensionalen Data Marts. (siehe dazu Kapitel 3.1) [Linstedt et al. 2015]

Das neue Data Vault Schema wird in Abbildung 17 illustriert. Dabei handelt es sich um ein vom Autor selbst modelliertes Schema, welches die einfachste Umsetzung des TPC-H Modells darstellt. Die Umsetzung in das Data Vault Schema erfolgte dabei in drei Schritten:

1. Die fachlichen Entitäten, also Objekte oder Ereignisse mit eigener Identität, wurden ermittelt und mittels eines Hubs dargestellt. Dabei handelt es sich einerseits um die Hubs Lieferant und Kunden (H_Supplier, H_Customer), andererseits um die Hubs Produkt und Bestellungen

(H_Part, H_Orders). Abbildung 14 illustriert die Projektion des Business Keys von der Tabelle „Supplier" in einen Hub.

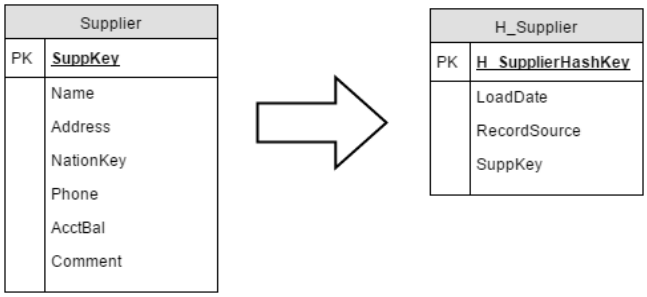


Abbildung 14: Projektion des Business Keys vom TPC-H Modell auf einen Hub

Um die vier Hubs zu erstellen, wurden die Business Objekte samt Business Keys im TPC-H Modell identifiziert. Für jedes Business Objekt entstand dann ein neuer Hub. Dieser beinhaltet neben dem Business Key auch die Datenquelle, in diesem Fall also die TPC-H Datenbank, und die Zeit des Ladens in das Data Vault. Weiterhin wird der Primärschlüssel aus dem Business Key errechnet. Die im TPC-H vorhandenen Tabellen „PartSupp" und „LineItem" sind keine eigenständigen Entitäten mit Identität und werden daher nicht als Hubs modelliert. Bei einem Lineitem handelt es sich lediglich um eine „Bestellzeile".

2. Die Beziehungen zwischen den Hubs werden mit Hilfe von Links modelliert. Der Link „L_SupplierNation" bildet die Beziehung zwischen den Lieferanten und einer Nation. Die fertige Modellierung aller zum Link gehörenden Hubs ist die Voraussetzung zur Umsetzung eines Links.

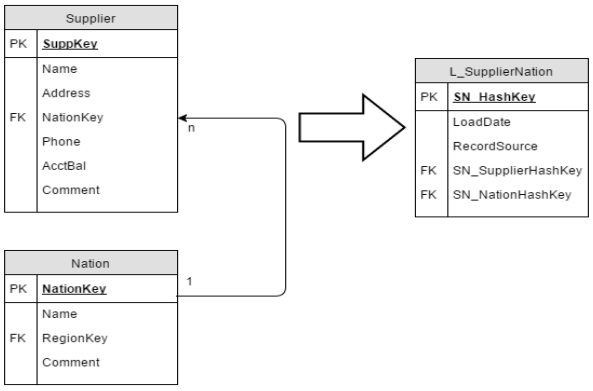


Abbildung 15: Projektion von Beziehungen auf einen Link

In Abbildung 15 wird anhand der Fremdschlüssel erkannt, dass eine Beziehung zwischen den Lieferanten und der Nation besteht. Um diese im Data Vault zu modellieren, müssen beide Entitäten bereits in Hubs existieren. Dann kann anhand beider Primärschlüssel (H_SupplierHashKey und NationKey) ein Link samt Hash-Schlüssel generiert werden. Mit

dem Load Date und der Record Source verhält es sich wie in Schritt 1 bei der Generierung von Hubs.

3. Die beschreibenden Attribute zu den Links und Hubs werden im letzten Schritt ermittelt und als Satelliten modelliert. Zwischen Satelliten und Hubs/Links besteht eine Many-to-one Beziehung, d.h. ein Satellit ist immer genau einem Hub oder Link zugeordnet, während ein Hub/Link mehrere Satelliten besitzen kann. Abbildung 16 zeigt die Projektion der beschreibenden Attribute der Tabelle „Supplier“ auf einen Satelliten.

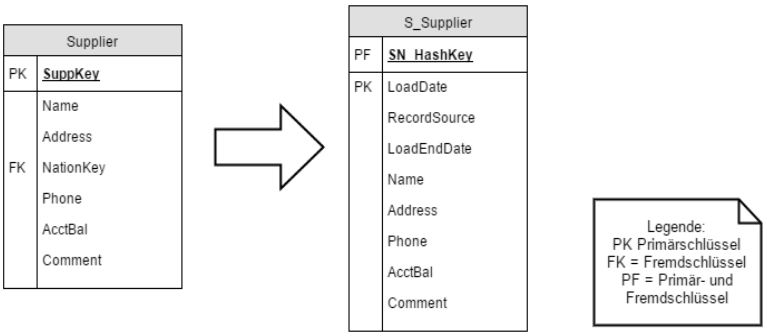


Abbildung 16: Projektion der beschreibenden Attribute auf einen Satelliten

Nachdem anhand des Primärschlüssels und des Fremdschlüssels ein Hub beziehungsweise ein Link modelliert werden konnte, werden nun die übrigen Attribute, welche die beschreibenden Attribute eines Objekts darstellen, als Satellit modelliert. Dabei wird aus dem für die Tabelle im Hub generierten Hash-Schlüssel und dem Zeitpunkt des Ladens der Primärschlüssel indiziert. Das LoadEndDate wird je nach Bedarf auf einen bestimmten Zeitpunkt in der Zukunft gesetzt. Die Record Source bildet wieder die TPC-H Datenbank.

Es fällt auf, dass das Data Vault Schema alle im TPC-H vorhandenen Entitäten, Attribute und Beziehungen zwischen den Entitäten übernommen und umgesetzt hat, während für die Umsetzung in das Starschema diverse Einschränkungen gemacht werden mussten. Dass Data Vault Modelle flexibel und erweiterbar sind, wurde schon in Abschnitt 6 ausreichend beschrieben. Im Hinblick auf den Vergleich mit dem Starschema wurde das Data Vault daher drastisch angepasst und so verkleinert, dass es dem Starschema vom Kontext her ähnlich ist. Abbildung 18 illustriert das stark verkleinerter Data Vault.

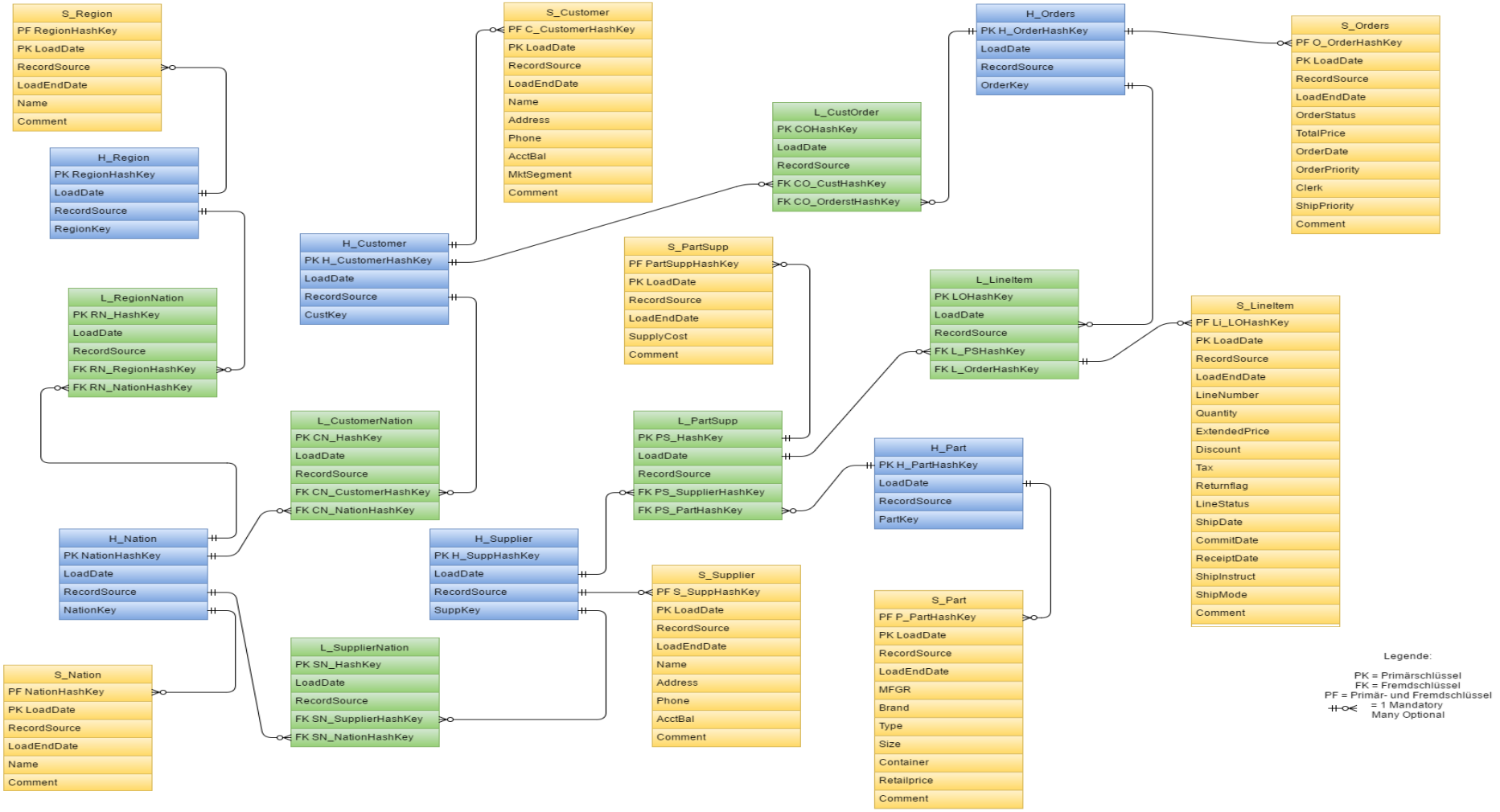


Abbildung 17: Data Vault Modell basierend auf dem TPC-H Modell

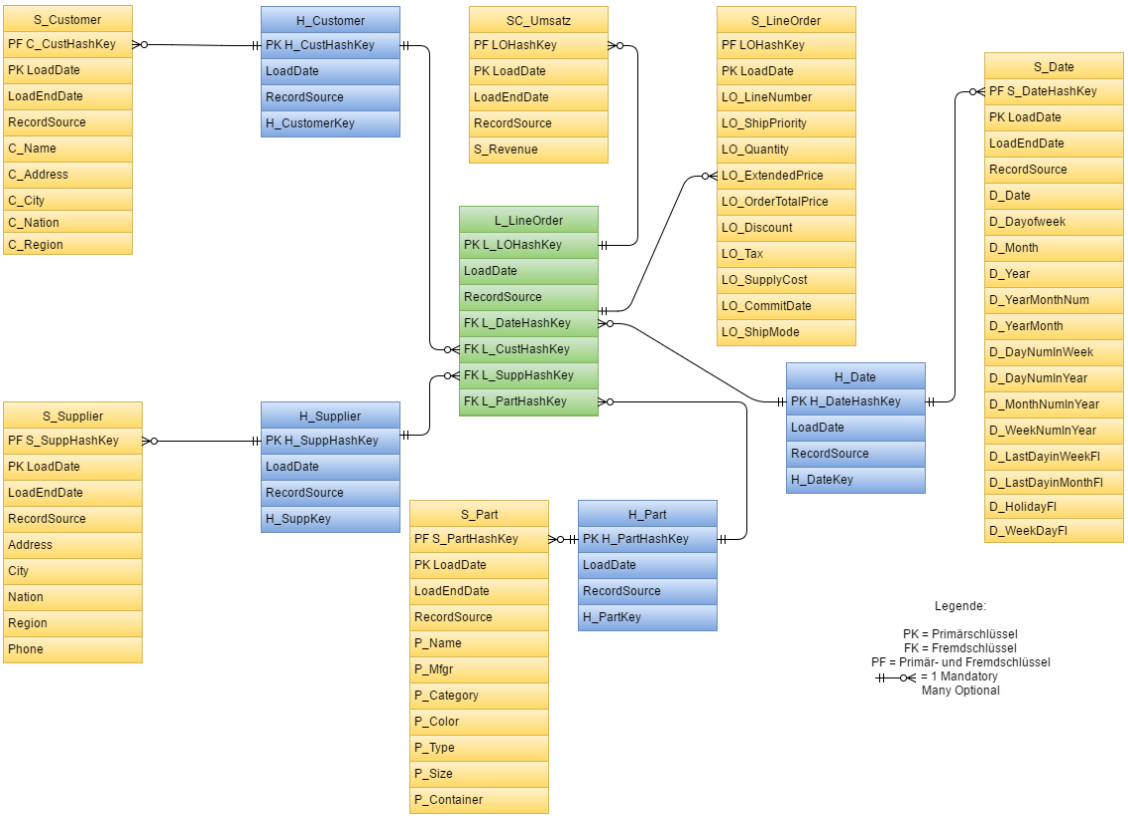


Abbildung 18: Data Vault Modell basierend auf dem SSB-Modell

Dieses Modell wurde wie das zuvor entwickelte Data Vault Modell nach den drei Schritten modelliert. Im Gegensatz zum ersten Data Vault, muss bei dem aus Abbildung 18 ein weiterer Aspekt bedacht werden. Die in der Faktentabelle des Starschemas neu eingefügte Kennzahl „Revenue“ muss durch einen extra Satelliten modelliert werden, der vom System generiert wird und die vor-aggregierten Werte enthält. Einen solchen Satelliten nennt man Computed Satellite. Computed Satellites wurden bereits in Abschnitt 3.2.3 beschrieben. Sie existieren nur im Business Vault. An das Attribut „Revenue“ sind eine Vielzahl von Abfragen im SSB gerichtet, sodass dieses letztendlich auch im Data Vault modelliert werden sollte. Somit findet kein Vergleich von Starschema und reinem Data Vault Modell statt, sondern von Starschema und einem im Business Vault adaptierten Modell.

8.3 Benchmark-Abfragen

Wie bereits beschrieben ist eine Eins-zu-eins-Anwendung der TPC-H Abfragen auf beide Datenmodelle nicht möglich. Der SSB von [O’Neil et al. 2009] nutzt daher größtenteils eigens definierte Abfragen, die nach dem typischen Data Warehouse Muster entwickelt wurden. Teilweise wurden aber auch Abfragen vom TPC-H Benchmark bedacht und angepasst, solange diese sich auf die Restriktion von Attributen innerhalb der Faktentabelle beschränkten. Bei der Erstellung eines für

den SSB optimierten, leistungsstarken Abfragepools wurden vor allem zwei Aspekte in den Vordergrund gestellt: die funktionale und die selektive Abdeckung.
Damit zukünftige Nutzer bei der Anwendung dieses Benchmarks auf den von ihnen in der Praxis eingesetzten Teilbereichen Performance-Messungen durchführen können, sollten die Abfragen den größtmöglichen funktionalen Bereich abdecken, d.h., dass die Menge an dimensionalen Restriktionen einer Abfrage so groß wie möglich gehalten werden sollte.
Weiterhin wurden die Abfragen so konzipiert, dass diese in ihren Filterfaktoren variieren. Filterfaktoren kalkulieren die Anzahl an Zeilen, die in die Ergebnisbearbeitung einbezogen werden. Die Variation derer beschreibt die selektive Abdeckung. Die unter Beachtung beider Aspekte entstandenen 13 Abfragen konnten in vier verschiedene Kategorien unterteilt werden. [O'Neil et al. 2009]
Die erste Kategorie K1 besitzt drei Abfragen, welche auf die Abfrage Q6 vom TPC-H aufbauen. Inhaltlich soll mit Hilfe dieser Abfragen ermittelt werden, inwiefern Umsatzsteigerungen durch die Eliminierung von bestimmten Rabatten für einen prozentualen Bereich an versendeten Produkten während eines bestimmten Zeitraums möglich gewesen wären. In den Abfragen Q1.1-Q1.3 werden demnach die Summen (ExtendedPrice * Discount) unter Slice-Funktionen bei der Quantität, des Discounts und der Zeit aggregiert. Da im SSB auf das Versanddatum (Shipdate) verzichtet wurde, wird anstelle dessen das Datum der Bestellung (OrderDate) verwendet. Die erste Klasse zeichnet sich auch dadurch aus, dass nur auf einer Dimension Beschränkungen erfolgen, nämlich der Zeit (C_Date).
Q1.1 filtert nach Bestelllinien (LineOrder) aus dem Jahr 1993, mit einem Discount zwischen 1 und 3, sowie einer Quantität kleiner 25:

```
SELECT sum(lo_extendedprice*lo_discount) as revenue
FROM lineorder, date where lo_orderdate = d_datekey
AND d_year = 1993
AND lo_discount between1 and 3
AND lo_quantity < 25;
```

Damit dieselbe Ergebnismenge wie beim Starschema erzielt werden kann, müssen die Abfragen für das Data Vault umgeschrieben werden. Da sowohl beim TPC-H als auch beim SSB in den Abfragen kartesische Produkte anstelle von Joins verwendet werden, werden auch die in den Data Vault Abfragen notwendigen Verknüpfungen durch kartesische Produkte erzeugt. Dies hat einerseits den Vorteil, dass die Abfrage an sich übersichtlicher dargestellt werden kann und an-

dererseits, dass der Vergleich der Abfragen von den verschiedenen Datenmodellen durch gleiche Syntax erleichtert wird. Um dies zu verdeutlichen wird im Folgenden die Abfrage Q1.1 aus dem SSB für das Data Vault Modell als Abfrage mit Join-Verknüpfung umgeschrieben:

```
SELECT sum(LO_ExtendedPrice*LO_Discount) AS revenue
FROM ((S_LineOrder JOIN L_LineOrder ON S_LineOrder.LOHashKey =
L_LineOrder.LOHashKey) JOIN H_Date ON H_Date.DateHashKey =
L_LineOrder.LO_DateHashKey) JOIN S_Date ON S_Date.DateHashKey = H_Date.DateHashKey
WHERE d_year = 1993
AND lo_discount between1 and 3
AND lo_quantity < 25;
```

Die Abfragen können durch diese Vielzahl an Joins schnell unübersichtlich werden. Im Vergleich dazu ist hier dieselbe Abfrage mit kartesischem Produkt:

```
SELECT sum(LO_ExtendedPrice*LO_Discount) AS revenue
FROM S_LineOrder, L_LineOrder, S_Date , H_Date
WHERE S_LineOrder.LOHashKey = L_LineOrder.L_LOHashKey
AND H_Date.H_DateHashKey = L_LineOrder.L_DateHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND d_year = 1993
AND lo_discount between1 and 3
And lo_quantity < 25;
```

Wie schon beim ersten quantitativen Vergleich in Abschnitt 5 festgestellt wurde, sind die Abfragen im Data Vault größer als im Starschema sind, da mehr Tabellen miteinander verknüpft werden müssen. Diese Erkenntnis wird sich beim Beschreiben der weiteren Abfragen in diesem Abschnitt verfestigen.

Q1.2 beschreibt kontextuell dieselbe Abfrage wie Q1.1, lediglich auf einer anderen zeitlichen Hierarchieebene. Anstelle aller Bestelllinien über das Jahr 1993 verteilt, filtert diese Abfrage nun nach allen Bestelllinien innerhalb eines Monats (Monat Januar im Jahr 1994). Des Weiteren wurden andere Filterwerte für den Discount (zwischen 4 und 6) und der Quantität (zwischen 26 und 35) gesetzt:

```
SELECT sum(lo_extendedprice*lo_discount) as revenue
FROM lineorder, date
WHERE lo_orderdate = d_datekey
AND d_yearmonthnum = 199401
AND lo_discount between4 and 6
AND lo_quantity between 26 and 35;
```

Wie für Q1.1 muss auch Q1.2 für das Data Vault angepasst werden:

```
SELECT sum(LO_ExtendedPrice*LO_Discount) AS revenue
FROM S_LineOrder, L_LineOrder, S_Date, H_Date
WHERE S_LineOrder.LOHashKey = L_LineOrder.L_LOHashKey
AND H_Date.H_DateHashKey = L_LineOrder.L_DateHashKey
AND S_Date.D_DateHashKey = H_Date.H_DateHashKey
AND d_yearmonthnum = 199401
AND lo_discount between4 and 6
AND lo_quantity between 26 and 35;
```

Die dritte Abfrage Q1.3 der ersten Kategorie filtert nach einer noch feineren Granularität der Zeitebene, indem alle Bestelllinien betrachtet werden, die sich auf eine bestimmte Woche (Woche 6 im Jahr 1994) erstrecken:

```
SELECT sum(lo_extendedprice*lo_discount) as revenue
FROM lineorder, date
WHERE lo_orderdate = d_datekey
AND d_weeknuminyear = 6
AND d_year = 1994
AND lo_discount between 5 and 7
AND lo_quantity between 26 and 35;
```

Das SSB definiert einen konstanten Wertebereich (numeric 1-53) für das Attribut Kalenderwoche(d_weeknuminyear), d.h. die erste Woche in jedem Jahr beginnt immer mit dem 1. Januar und endet mit dem 6. Januar, die zweite Woche besteht somit aus den nächsten sieben Tagen,

und so weiter. Demnach besteht jede Kalenderwoche aus sieben Tagen, während in der Realität Woche 1 und Woche 53 eines Jahres aus einer Menge von 1-7 Tagen bestehen kann und somit die einzelnen Daten einer Kalenderwoche jährlich voneinander abweichen können. Deshalb muss bei diesen Abfragen bedacht werden, keine vom DBMS bereitgestellten Funktionen zum Errechnen der Kalenderwochen zu nutzen. Als Beispiel sei hier die DatePart-Funktion genannt. Während für Kalenderwoche 6 im SSB die Daten pro Jahr konstant zwischen dem 04.02.yyyy und dem 10.02.yyyy liegen, würde die DatePart-Funktion für jedes Jahr eine unterschiedliche Datenspanne ausgeben. Da jedoch auch im Data Vault die Tabelle „Date“ existiert, kann die Ddata Vault Abfrage wie folgt aussehen:

*SELECT sum(LO_ExtendedPrice*LO_Discount) AS revenue*
FROM S_LineOrder, L_LineOrder, S_Date, H_Date
WHERE S_LineOrder.LOHashKey = L_LineOrder.L_LOHashKey
AND H_Date.H_DateHashKey = L_LineOrder.L_DateHashKey
AND S_Date.D_DateHashKey = H_Date.H_DateHashKey
AND d_weeknuminyear = 6
AND d_year = 1994
AND lo_discount between 5 and 7
AND lo_quantity between 26 and 35;

Alle Abfragen der Kategorie 1 dienen inhaltlich demselben Zweck, allerdings auf unterschiedlichen Granularitätsebenen der Zeit (Jahr → Monat → Woche). Die Änderungen der Filterwerte von den Attributen Quantität und Discount sind dem geschuldet, dass dadurch bei der Ausführung aller drei Abfragen keine Überlappungen der verknüpften Mengen in der Faktentabelle entstehen, wodurch der Caching-Effekt reduziert wird.
Kategorie 2 beinhaltet wiederum auch drei Abfragen. Diese Abfragen sollen weitere typische kontextbezogene Data Warehouse Abfragen abdecken, welche inhaltlich jedoch nicht, im Gegensatz zu Kategorie 1, im TPC-H zu finden sind. Die Abfragen vergleichen bestimmte Produktklassen aus einer bestimmten Lieferregion anhand des Umsatzes und der Marken, für alle Jahre in denen diese bestellt wurden. Diese Abfragen entsprechen jener eines typischen Data Warehouses, bestehend aus einer Select-Klausel mit Aggregation (sum(lo_revenue)), einer From-Klausel, einer Where-Klausel mit Restriktionen (u.a. s_region = 'AMERICA') und einer Group-By Funktion. Während in Kategorie 1 Restriktionen in einer Dimension erfolgten, werden nun zwei Dimensionen beschränkt.

Die Abfrage Q2.1 sucht alle Marken und deren erzielten summierten Umsatz pro Jahr für die Produktkategorie 'MFGR#12', welche aus Amerika geliefert wurden:

```
SELECT sum(Lo_Revenue), d_year, p_brand
FROM lineorder, date, part, supplier
WHERE lo_orderdatekey = d_datekey
AND lo_partkey = p_partkey
AND lo_suppkey = s_suppkey
AND p_category = 'MFGR#12'
AND s_region = 'AMERICA'
GROUP BY d_year, p_brand
ORDER BY d_year, p_brand;
```

Auf das Data Vault angewendet sieht die Abfrage dementsprechend wie folgt aus:

```
SELECT sum(lo_revenue), D_year, P_brand
FROM S_Umsatz, L_LineOrder, H_Date, H_Part, H_Supplier S_Date, S_Part, S_Supplier
WHERE S_Umsatz.L_LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_Date.H_DateHashKey
AND L_LineOrder.L_PartHashKey = H_Part.H_PartHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Part.H_PartHashKey = S_Part.PartHashKey
AND H_Supplier.H_SuppHashKey = S_Supplier.S_SuppHashKey
AND P_category = 'MFGR#12'
AND S_Region = 'AMERICA'
GROUP BY D_Year, P_Brand
ORDER BY D_year, P_brand;
```

Die Abfrage Q2.2 beschränkt die Region dieses Mal auf Asien und ersetzt die Produktkategorie in der Where-Klausel durch die Produktmarken 'MFGR#2221' und 'MFGR#2228', um auf einer anderen Hierarchieebene zu arbeiten. Somit werden nicht alle Umsätze für eine bestimmte Produktkategorie generiert, sondern für die zwei Produktmarken:

```
SELECT sum(lo_revenue), d_year, p_brand
FROM lineorder, date, part1, supplier
```

```
WHERE lo_orderdatekey = d_datekey
AND lo_partkey = p_partkey
AND lo_suppkey = s_suppkey
AND p_brand between 'MFGR#2221' and 'MFGR#2228'
AND s_region = 'ASIA'
GROUP BY d_year, p_brand
ORDER BY d_year, p_brand;
```

Die Data Vault Abfrage verändert sich kaum gegenüber der von Q2.1, nur die besagten Restriktionen werden aktualisiert:

```
SELECT sum(lo_revenue), D_year, P_brand
FROM S_Umsatz, L_LineOrder, H_Date, H_Part, H_Supplier S_Date, S_Part, S_Supplier
WHERE S_Umsatz.L_LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_Date.H_DateHashKey
AND L_LineOrder.L_PartHashKey = H_Part.H_PartHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Part.H_PartHashKey = S_Part.PartHashKey
AND H_Supplier.H_SuppHashKey = S_Supplier.S_SuppHashKey
AND P_Brand between 'MFGR#2221' and 'MFGR#2228'
AND s_region = 'ASIA'
GROUP BY D_Year, P_Brand
ORDER BY D_year, P_brand;
```

Die dritte Abfrage Q2.3 ersetzt die beiden Produktmarken in der Where-Klausel durch eine einzige Produktmarke ‘MFGR#2339’ und beschränkt die Lieferantenregion von Asien auf Europa:

```
SELECT sum(lo_revenue), d_year, p_brand
FROM lineorder, date, part1, supplier
WHERE lo_orderdatekey = d_datekey
AND lo_partkey = p_partkey
AND lo_suppkey = s_suppkey
AND p_brand = 'MFGR#2339'
AND s_region = 'EUROPE'
```

```
GROUP BY d_year, p_brand
ORDER BY d_year, p_brand;
```

Q2.3 im Data Vault sieht demnach wie folgt aus:

```
SELECT sum(lo_revenue), D_year, P_brand
FROM S_Umsatz, L_LineOrder, H_Date, H_Part, H_Supplier S_Date, S_Part, S_Supplier
WHERE S_Umsatz.L_LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_Date.H_DateHashKey
AND L_LineOrder.L_PartHashKey = H_Part.H_PartHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Part.H_PartHashKey = S_Part.PartHashKey
AND H_Supplier.H_SuppHashKey = S_Supplier.S_SuppHashKey
AND P_Brand = 'MFGR#2339'
AND S_Region = 'EUROPE'
GROUP BY D_Year, P_Brand
ORDER BY D_year, P_brand;
```

Kategorie 3 besteht aus vier Abfragen, welche sowohl ein Drill-Down als auch Roll-Up auf der Kunden- bzw. Lieferantentabelle ausüben. Dabei werden die hierarchischen Gegebenheiten der Attribute City → Nation → Region beider Tabellen ausgenutzt. Die Abfragen basieren auf der Abfrage Q5 des TPC-H Benchmarks und besitzen nun Restriktionen auf drei Dimensionstabellen. Inhaltich soll mit dieser Kategorie das Umsatzvolumen von Bestelllinien pro Nation des Kunden und des Lieferanten pro Jahr ermittelt werden, wobei dies auf eine bestimmte Region und Zeitperiode beschränkt wird.
Q3.1 beschränkt sich auf die Kunden- und Lieferantenregion Asien und einer Zeitperiode von 6 Jahren zwischen 1992 und 1997:

```
SELECT c_nation, s_nation, d_year, sum(lo_revenue) as revenue
FROM customer, lineorder, supplier, date
WHERE lo_custkey = C_CustomerKey
AND lo_suppkey = s_suppkey
AND lo_orderdatekey = d_datekey
AND c_region = 'ASIA' and s_region = 'ASIA'
```

```
AND d_year >= 1992 and d_year <= 1997
GROUP BY c_nation, s_nation, d_year
ORDER BY d_year asc, revenue desc;
```

Die dazugehörige Data Vault Abfrage sieht wie folgt aus:

```
SELECT C_Nation, S_Nation, D_Year, sum(LO_Revenue) as revenue
FROM S_Umsatz, L_LineOrder, H_Date, H_Supplier S_Date, S_Supplier, S_Customer,
H_Customer
WHERE S_Umsatz.L_LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_Date.H_DateHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND L_LineOrder.L_CustHashKey = H_Customer.H_CustHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Supplier.H_SuppHashKey = S_Supplier.S_SuppHashKey
AND H_Customer.H_CustHashKey = S_Customer.C_CustHashKey
AND C_Region = 'ASIA'
AND S_ Region = 'ASIA'
AND d_year >= 1992 and d_year <= 1997
GROUP BY C_ Nation, S_ Nation, D_Year
ORDER BY D_Year asc, LO_revenue desc;
```

In Q3.2 sollen alle aus der Nation USA kommenden Kunden und Lieferanten bedacht werden. Die Zeitperiode bleibt die gleiche:

```
SELECT c_city, s_city, d_year, sum(lo_revenue) as revenue
FROM customer, lineorder, supplier, date
WHERE lo_custkey = C_CustomerKey
AND lo_suppkey = s_suppkey
AND LO_OrderDateKey = d_datekey
AND c_nation = 'UNITED STATES'
AND s_nation = 'UNITED STATES'
AND d_year >= 1992 and d_year <= 1997
GROUP BY c_city, s_city, d_year
ORDER BY d_year asc, revenue desc;
```

Die Abfrage für das Data Vault lautet:

```
SELECT C_City, S_City, D_Year, sum(LO_Revenue) as revenue
FROM S_Umsatz, L_LineOrder, H_Date, H_Supplier S_Date, S_Supplier, S_Customer,
H_Customer
WHERE S_Umsatz.L_LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_Date.H_DateHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND L_LineOrder.L_CustHashKey = H_Customer.H_CustHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Supplier.H_SuppHashKey = S_Supplier.S_SuppHashKey
AND H_Customer.H_CustHashKey = S_Customer.C_CustHashKey
AND c_nation = 'UNITED STATES'
AND s_nation = 'UNITED STATES'
AND d_year >= 1992 and d_year <= 1997
GROUP BY C_ City, S_ City, D_Year
ORDER BY D_Year asc, LO_revenue desc;
```

Abfrage Q3.3 verfeinert die Granularität und beschränkt die Ausgabe nach den jeweiligen Kunden- und Lieferanten-Städten UNITED KI1 oder UNITED KI2. Auch hier bleibt die Zeitperiode konstant:

```
SELECT c_city, s_city, d_year, sum(lo_revenue) as revenue
FROM customer, lineorder, supplier, date
WHERE lo_custkey = C_CustomerKey
AND lo_suppkey = s_suppkey
AND LO_OrderDateKey = d_datekey
AND (c_city='UNITED KI1' or c_city='UNITED KI5')
AND (s_city='UNITED KI1' or s_city='UNITED KI5')
AND d_year >= 1992 and d_year <= 1997
GROUP BY c_city, s_city, d_year
ORDER BY d_year asc, revenue desc;
```

Im Data Vault sieht die Abfrage wie folgt aus:

```
SELECT C_City, S_City, D_Year, sum(LO_Revenue) as revenue
FROM S_Umsatz, L_LineOrder, H_Date, H_Supplier S_Date, S_Supplier, S_Customer,
H_Customer
WHERE S_Umsatz.L_LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_Date.H_DateHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND L_LineOrder.L_CustHashKey = H_Customer.H_CustHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Supplier.H_SuppHashKey = S_Supplier.S_SuppHashKey
AND H_Customer.H_CustHashKey = S_Customer.C_CustHashKey
AND (c_city='UNITED KI1' or c_city='UNITED KI5')
AND (s_city='UNITED KI1' or s_city='UNITED KI5')
AND d_year >= 1992 and d_year <= 1997
GROUP BY C_ City, S_ City, D_Year
ORDER BY D_Year asc, LO_revenue desc;
```

Die vierte Abfrage Q3.4 bezieht sich auf dieselben Beschränkungen der Städte, jedoch wird hier die Date-Dimension auf Dezember im Jahr 1997 gesetzt:

```
SELECT c_city, s_city, d_year, sum(lo_revenue) as revenue
FROM customer, lineorder, supplier, date
WHERE lo_custkey = C_CustomerKey
AND lo_suppkey = s_suppkey
AND LO_OrderDateKey = d_datekey
AND (c_city='UNITED KI1' or c_city='UNITED KI5')
AND (s_city='UNITED KI1' or s_city='UNITED KI5')
AND d_yearmonth = 'Dec1997'
GROUP BY c_city, s_city, d_year
ORDER BY d_year asc, revenue desc;
```

Für das Data Vault lautet Abfrage Q3.4 demnach:

```
SELECT C_City, S_City, D_Year, sum(LO_Revenue) as revenue
FROM S_Umsatz, L_LineOrder, H_Date, H_Supplier S_Date, S_Supplier, S_Customer,
H_Customer
WHERE S_Umsatz.L_LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_Date.H_DateHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND L_LineOrder.L_CustHashKey = H_Customer.H_CustHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Supplier.H_SuppHashKey = S_Supplier.S_SuppHashKey
AND H_Customer.H_CustHashKey = S_Customer.C_CustHashKey
AND (c_city='UNITED KI1' or c_city='UNITED KI5')
AND (s_city='UNITED KI1' or s_city='UNITED KI5')
AND d_yearmonth = 'Dec1997'
GROUP BY C_ City, S_ City, D_Year
ORDER BY D_Year asc, LO_revenue desc;
```

Kategorie 4 setzt sich aus drei Abfragen zusammen, die ein Drill-Down bzw. Roll-Up auf Spalten der Tabellen Part, Customer und Supplier ausführen. Die Abfragen dieser Kategorie befassen sich mit dem Errechnen des Profits, welcher aus der Differenz von Umsatz und Lieferkosten ermittelt werden kann. Weiterhin sollen das Jahr und die Nation des Kunden für den jeweiligen Umsatz ausgegeben werden. Beschränkungen finden sowohl über die Region von Kunde und Lieferant statt, als auch über die Manufakturen der Lieferanten. Die Definition und Anordnung der Abfragen erfolgte nach dem What-if Ansatz, bei den sich anhand der Ausgaben der Abfragen und deren Interpretation durch den Anwender neue Abfragen bilden.
Q4.1 setzt die Regionen auf Amerika und die Manufakturen als Auswahl von MFGR#1 oder MFGR#2:

```
SELECT d_year, c_nation, sum(lo_revenue -lo_supplycost) as profit
FROM date, customer, supplier, part, lineorder
WHERE lo_custkey = C_CustomerKey
AND lo_suppkey = s_suppkey
AND lo_partkey = p_partkey
AND LO_OrderDateKey = d_datekey
AND c_region = 'AMERICA'
```

```
AND s_region = 'AMERICA'
AND (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
GROUP BY d_year, c_nation
ORDER BY d_year, c_nation;
```

Die dazugehörige Data Vault Abfrage lautet:

```
SELECT d_year, c_nation, sum(lo_revenue -lo_supplycost) as profit
FROM S_Umsatz, L_LineOrder, H_Date, H_Supplier S_Date, S_Supplier, S_Customer,
H_Customer, S_LineOrder, H_Part, H_Date
WHERE S_Umsatz.L_LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_Date.H_DateHashKey
AND L_LineOrder.L_PartHashKey = H_PartHashKey
AND S_LineOrder.LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND L_LineOrder.L_CustHashKey = H_Customer.H_CustHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Supplier.H_SuppHashKey = S_Supplier.S_SuppHashKey
AND H_Customer.H_CustHashKey = S_Customer.C_CustHashKey
AND H_Part.H_PartHashKey = S_Part.P_PartHashKey
AND c_region = 'AMERICA'
AND s_region = 'AMERICA'
AND (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
GROUP BY d_year, c_nation
ORDER BY d_year, c_nation;
```

In der Annahme, dass die Abfrage eine enorme Steigerung des Profits in den Jahren 1997 und 1998 aufzeigt (dieses muss in der Datenbank nicht der Wahrheit entsprechen und dient nur als Beispiel für den What-if Ansatz), wurde die Where-Klausel in Q4.2 zusätzlich mit einer Restriktion auf die Jahreszahlen erweitert. Des Weiteren soll mit Hilfe von Q4.2 herausgefunden werden, auf welche Produktkategorien und welche Nation sich dieser Profit in den beiden in Q4.1 ermittelten Jahren erstreckt:

```
SELECT d_year, s_nation, p_category, sum(lo_revenue - lo_supplycost) as profit
FROM date, customer, supplier, part, lineorder
WHERE lo_custkey = C_CustomerKey
```

```
AND lo_suppkey = s_suppkey
AND lo_partkey = p_partkey
AND LO_OrderDateKey = d_datekey
AND c_region = 'AMERICA'
AND s_region = 'AMERICA'
AND (d_year = 1997 or d_year = 1998)
AND (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
GROUP BY d_year, s_nation, p_category
ORDER BY d_year, s_nation, p_category;
```

Die darauf aufbauende Data Vault Abfrage sieht wie folgt aus:

```
SELECT d_year, s_nation, p_category, sum(lo_revenue -lo_supplycost) as profit
FROM S_Umsatz, L_LineOrder, H_Date, H_Supplier S_Date, S_Supplier, S_Customer,
H_Customer, S_LineOrder, H_Part, H_Date
WHERE S_Umsatz.L_LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_Date.H_DateHashKey
AND L_LineOrder.L_PartHashKey = H_PartHashKey
AND S_LineOrder.LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND L_LineOrder.L_CustHashKey = H_Customer.H_CustHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Supplier.H_SuppHashKey = S_Supplier.S_SuppHashKey
AND H_Customer.H_CustHashKey = S_Customer.C_CustHashKey
AND H_Part.H_PartHashKey = S_Part.P_PartHashKey
AND c_region = 'AMERICA'
AND s_region = 'AMERICA'
AND (d_year = 1997 or d_year = 1998)
AND (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
GROUP BY d_year, s_nation, p_category
ORDER BY d_year, s_nation, p_category;
```

Abfrage Q4.3 basiert nun auf der Annahme, dass der Großteil des Profits in den USA durch die Produktkategorie MFGR#14 generiert wurde. Um einen weiteren Drill-Down auf Stadtebene und Produktmarke analysieren zu können, lautet die folgende Abfrage wie folgt:

```
SELECT d_year, s_city, p_brand, sum(lo_revenue-lo_supplycost) as profit
FROM date, customer, supplier, part, lineorder
WHERE lo_custkey = C_CustomerKey
AND lo_suppkey = s_suppkey
AND lo_partkey = p_partkey
AND LO_OrderDateKey = d_datekey
AND c_region = 'AMERICA'
AND s_nation = 'UNITED STATES'
AND (d_year = 1997 or d_year = 1998)
AND p_category = 'MFGR#14'
GROUP BY d_year, s_city, p_brand
ORDER BY d_year, s_city, p_brand;
```

Die Umsetzung der Abfrage im Data Vault ist definiert als:

```
SELECT d_year, s_city, p_brand, sum(lo_revenue -lo_supplycost) as profit
FROM S_Umsatz, L_LineOrder, H_Date, H_Supplier S_Date, S_Supplier, S_Customer,
H_Customer, S_LineOrder, H_Part, H_Date
WHERE S_Umsatz.L_LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_Date.H_DateHashKey
AND L_LineOrder.L_PartHashKey = H_PartHashKey
AND S_LineOrder.LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND L_LineOrder.L_CustHashKey = H_Customer.H_CustHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Supplier.H_SuppHashKey = S_Supplier.S_SuppHashKey
AND H_Customer.H_CustHashKey = S_Customer.C_CustHashKey
AND H_Part.H_PartHashKey = S_Part.P_PartHashKey
AND c_region = 'AMERICA'
AND s_nation = 'UNITED STATES'
AND (d_year = 1997 or d_year = 1998)
AND p_category = 'MFGR#14'
GROUP BY d_year, s_city, p_brand
ORDER BY d_year, s_city, p_brand;
```

Anhand aller Abfragen kann man die Schlüsse ziehen, dass die Definition und Bearbeitung derer im Data Vault (DV) wesentlich aufwendiger ist. Während man beim Starschema durch eine Faktentabelle, welche mit allen Dimensionstabellen direkt in Beziehung steht, relativ wenige Verknüpfungen braucht, ist durch die separate Behandlung von beschreibenden Objekten, deren Business Keys und deren Beziehungen zu anderen Objekten durch Satelliten, Hubs und Links eine höhere Anzahl an Verknüpfungen nötig. In Tabelle 2 wird deutlich, dass die Anzahl der verknüpften Tabellen im Data Vault Schema noch überschaubar ist, solange Restriktionen auf wenigen Dimensionen im Starschema stattfinden. Bei Restriktionen auf vier Dimensionstabellen steigt die Anzahl an Verknüpfungen beim Data Vault in den zweistelligen Bereich.
Die Filterfaktoren unterscheiden sich in beiden Schemata nicht voneinander. Die Restriktionen der Abfragen werden im Data Vault größtenteils auf den Satelliten ausgeführt, da diese die beschreibenden Attribute eines Objekts oder Links besitzen.
Tabelle 2 zeigt auch auf, dass sich der Pool aus dreizehn Abfragen als guter Ansatz zur Performance-Messung erweist, da er sowohl Abfragen beinhaltet, welche auf Basis des TPC-H Benchmarks definiert wurden, als auch What-If Abfragen. Kategorie 2 konnte keinem der beiden Ansätze zugeordnet werden. Durch die auf dem TPC-H Benchmark basierenden Abfragen erhält die Performance-Messung eine solide Objektivität eines anerkannten Benchmarks zur Messung von Datenbankperformance. Die What-If Abfragen bringen wiederum subjektiv wahrgenommene hypothetische Szenarien aus dem Alltag mit, welche für die Nutzer von Decision Support Systemen von großer Bedeutung sind. [Xu, Huan and Luo, Hao and He, Jieyue 2013]

Tabelle 2: Vergleich SSB und Data Vault Abfragen

	Grundlage		**Anzahl verknüpfter Tabellen**		**Anzahl an Slice**		**Anzahl Tabellen, auf denen Restriktionen laufen**		**Group-By Klausel**	**Filter Faktor**	
	TPC-H	What-IF Ansatz	SSB	DV	SSB	DV	SSB (ohne Faktentabelle)	DV		SSB (auf Dimensionen)	DV (auf Satelliten)
Q1.1	X		2	4	3	3	1	2		0,019	0,019
Q1.2	X		2	4	3	4	1	2		0,00065	0,00065
Q1.3	X		2	4	4	4	1	2		0,000065	0,000065
Q2.1		.	4	8	2	2	2	2	X	0,0080	0,0080
Q2.2			4	8	2	2	2	2	X	0,0016	0,0016
Q2.3			4	8	2	2	2	2	X	0,00020	0,00020
Q3.1	X		4	8	4	4	3	3	X	0,036	0,036
Q3.2	X		4	8	4	4	3	3	X	0,0014	0,0014
Q3.3	X		4	8	4	4	3	3	X	0,000055	0,000055
Q3.4	X		4	8	3	4	3	3	X	0,00000076	0,00000076
Q4.1		X	5	11	3	3	3	3	X	0,016	0,016
Q4.2		X	5	11	4	4	4	4	X	0,0046	0,0046
Q4.3		X	5	11	4	4	4	4	X	0,000091	0,000091

8.4 Einrichtung der Testumgebung

Zur Performance-Messung wird dieselbe Hard- und Software genutzt, welche auch bei der ersten Test-Messung in Abschnitt 4 verwendet wurde:

- Mainboard: Lenovo 3259HLG
- Prozessor: Intel® Core™ i5-3210M CPU @ 2.50GHz, 2501 MHz, 2 Kern(e), 4 logische Prozessoren
- Arbeitsspeicher: 4GB (DDR3 SDRAM)
- Festplatte: 500 GB SATA II 3.0Gb/s (2.5") Festplatte mit 7.200 Umdrehungen pro Minute (upm)
- Solid State Drive: -
- Laufwerk: CD / DVD-ROM Laufwerk
- Netzteil: Original Lenovo 65 Watt 20 V Netzteil
- Betriebssystem: Microsoft Windows 10 Pro
- Software: Microsoft SQL Server

Erstellt wurden die Daten für das Starschema-Modell mit dem vom SSB bereitgestellten Datengenerator, der eine modifizierte DBGEN-Anwendung vom TPC-H darstellt. Insgesamt kommt man somit bei Auswahl des Skalenfaktors 1 auf eine Gesamtgröße von rund 4,5 GB. Tabelle 3 gibt einen Überblick über die Anzahl der Datensätze pro Tabelle.

Tabelle 3: Anzahl an Datensätzen im SSB

Tabelle	Anzahl Datensätze
LineOrder	6001215
Customer	30000
Supplier	10000
Part	200000
Date	2556
	∑6243771

Das Data Vault Modell wurde so modelliert, dass alle Datensätze des SSBs integriert werden konnten. Dabei wurden die für das Data Vault Schema üblichen Komponenten bestehend aus Satelliten, Hubs und Links erstellt und mit den Datensätzen vom SSB beladen. Aufgrund der größeren Anzahl an Tabellen und des Integrierens der Hash-Schlüssel kommt das Data Vault Modell auf eine Größe von ungefähr 13 GB und eine vergleichsweise hohe Anzahl an Datensät-

zen, die sich dank der Einzelbetrachtung von beschreibenden Attributen und Business Keys im Data Vault fast verdreifacht hat. (siehe Tabelle 4)

Tabelle 4: Anzahl der Datensätze im Data Vault

Tabelle	Anzahl Datensätze
H_Customer	30000
H_Supplier	10000
H_Part	200000
H_Date	2556
S_Customer	30000
S_Supplier	10000
S_Date	2556
S_Part	200000
L_LineItem	6001215
S_LineItem	6001215
S_Umsatz	6001215
	∑18488757

8.5 Erstellung des Abfrageplans

Das TPC gibt Vorgaben in Bezug auf die Durchführung einer Performance-Messung von Datenbankmanagementsystemen. Zum einen verlangt es die Durchführung eines Power-Tests, in welchem eine Abfragesequenz innerhalb einer Nutzersession durchgeführt wird. Die Abfragesequenz besteht aus drei Teilen, die in folgender Reihenfolge abgearbeitet werden müssen:

- Aktualisierungsprozess zum Einfügen neuer Datensätze (Insert)
- Abfrage-Templates
- Aktualisierungsprozess zum Löschen von Datensätzen (Delete)

Zum anderen soll ein Throughput-Test die Nebenläufigkeit von Systemen testen, indem man Abfragesequenzen und Aktualisierungsprozesse parallel durch mehrere Nutzer laufen lässt. Aufgrund der Tatsache, dass beide Datenmodelle in dieser Arbeit auf demselben Datenbanksystem laufen und von nur einem Nutzer bedient werden, erübrigt sich bei diesem Vergleich der Throughput-Test. Um den Bezug zum TPC-H Benchmark aufrechtzuerhalten, wurde daher im Folgenden ein Power-Test auf beide Datenmodelle durchgeführt. [TPC 2014]

Als erstes wird innerhalb einer Anfragesequenz der Aktualisierungsprozess zum Einfügen neuer Datensätze durchgeführt. Im SSB bietet sich die Möglichkeit an, Skripte zum Einfügen neuer Datensätze in die Tabelle „LineOrder“ automatisch generieren zu lassen. Für den Skalenfaktor 1 sind dies knapp 6000 Datensätze. Um die gleiche Anzahl an Datensätzen in das Data Vault zu laden, müssen diese Skripte jedoch angepasst werden. Zieht man sich zur Verdeutlichung die AbbildungenAbbildung 13 und Abbildung 18 heran, erkennt man, dass die LineOrder-Tabelle aus dem SSB im Data Vault Modell auf drei Tabellen umgesetzt wurde. Um alle Attribute eines Line-Order-Objekts einzufügen, mussten dementsprechend drei Skripte für das Data Vault generiert werden, die mittels BULK-Load in die Datenbank geladen wurden.

Der zweite Teil einer Anfragesequenz im Power-Test besteht aus dem Hintereinanderschalten der Abfragen. Hier wurden die in Abschnitt 8.3 definierten Abfragen für das jeweilige Datenmodell herangezogen und nacheinander in der Reihenfolge, in der sie beschrieben wurden, ausgeführt. Zwischen jeder Abfrage wurde mittels SQL-Statement der Cache und Puffer des SQL Servers geleert, um bereits benannte Caching-Effekte zu vermeiden.

Das Ende einer Anfragesequenz im Power-Test besteht aus einem Aktualisierungsprozess, um vorhandene Datensätze zu löschen. In der in diesem Vergleich umgesetzten Sequenz wurde letztlich eine Abfrage geschrieben, die die in Teil 1 einfügten Datensätze anhand ihres Ladeda-tums löschen. Abbildung… im Anhang zeigt die Anfragesequenz für den Power-Test in der Data Vault Implementierung.

Als Leistungsindikatoren für die Messung der Performance wurden die Laufzeit (Elapsed Time), d.h. die Zeit gemessen von der Ausführung der Abfrage bis zum Ausgeben deren Ergebnismen-ge, sowie die Prozessorzeit (CPU Time) der einzelnen Abfragen festgelegt.

8.6 Performance-Messung

Abbildung 19 illustriert die Laufzeiten der einzelnen Abfragen angewendet auf die verschiedenen Datenmodelle. Die Abfragesequenzen wurden sowohl für das Data Vault als auch das Starsche-ma in zeilenorientierter Speicherung und auch spaltenorientierter Speicherung, umgesetzt durch Column Store Indizes, angewendet.

Es ist deutlich zu erkennen, dass die spaltenorientierte Speicherung Performance-Vorteile ge-genüber der zeilenorientierten Speicherung hat. Dies liegt zum einen an der Beschränkung dieser Indizes, keine weiteren Indizes wie Primär- und Fremdschlüssel zu erlauben. Zum anderen wer-den die Vorteile von Column Stores in dieser Abbildung deutlich sichtbar. Da die einzelnen Spal-ten der Tabellen im SSB und Data Vault häufig gleiche Daten aufweisen (die Spalte Discount besteht zum Beispiel durchgehend aus den Zahlen 0-10), können hohe Komprimierungsraten

erzielt werden, welche sich in der Abfrageleistung positiv wiederspiegeln. Weiterhin werden bei den Abfragen des SSB vergleichsweise wenige Spalten einer Tabelle ausgewählt, wo durch die Anzahl der I/O-Vorgänge auf das physische Medium reduziert wird. [Abadi et al. 2009]

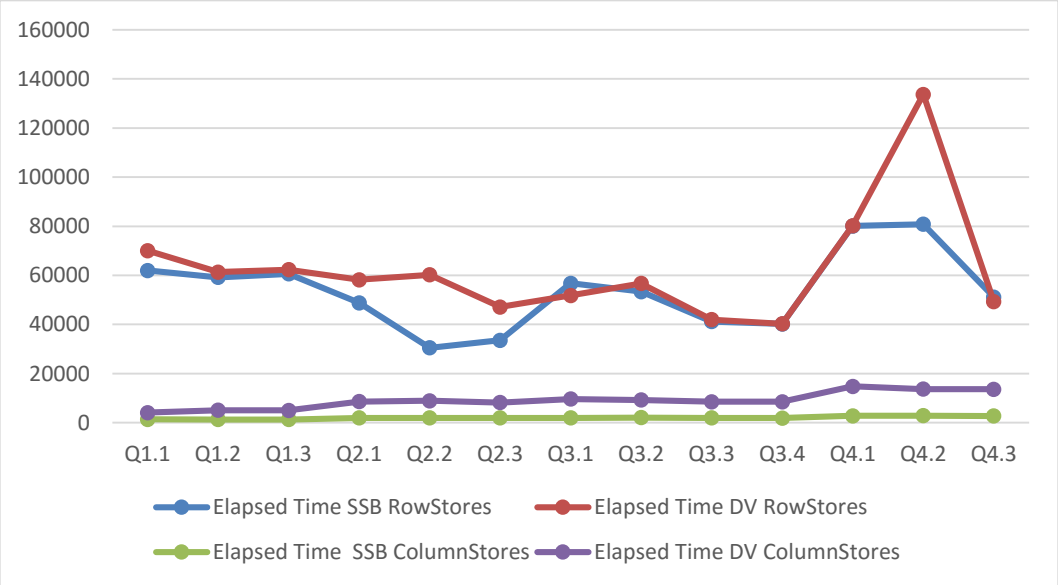


Abbildung 19: Laufzeit der Abfragen bezogen auf den verschiedenen Datenmodellen

Während bei den Column Store Varianten eine relativ lineare Zunahme der Laufzeit in einer Abfragesequenz ersichtlich wird, gibt es bei den zeilenorientierten Varianten Schwankungen sowohl nach oben als nach unten. Das Data Vault Modell performt zusammengenommen wahrscheinlich auch wegen der größeren Anzahl an Verknüpfungen zwischen Tabellen minimal schlechter als das Starschema. Ein Grund dafür könnten die verwendeten Primär- und Fremdschlüssel sein. Während die Entitäten im Starschema durch künstlich erzeugte Surrogat-Schlüssel mit unterschiedlicher Länge identifiziert werden, kommen im Data Vault die Hash-Schlüssel zum Einsatz. Die konstant gleichbleibende Länge dieser Schlüssel kann bei Vergleichen zwischen Datensätzen und Tabellen zu schnelleren Ergebnissen führen. Jedoch tritt dieser Effekt erst bei größeren Datenmengen auf, bei denen die Schlüssel entweder aus unterschiedlichen Datentypen bestehen oder aber eine größere Länge als der Hash-Schlüssel aufweisen. [Linstedt et al. 2015] Die separate Behandlung von Business Keys und beschreibenden Attributen in Hubs und Satelliten fällt in diesem Vergleich noch nicht so sehr ins Gewicht, solange das Data Vault Modell wie in diesem Beispiel klein gehalten ist.

Im Gegensatz zum Starschema ist die CPU-Nutzung beim Data Vault höher, was man anhand der Prozessorzeit in Abbildung 20 deutlich sehen kann. Die Column Store Varianten schneiden aufgrund deren spaltenweisen Speicherung jeweils schlechter ab als die Varianten der zeilenorientierten Speicherung, da die Kompressions- beziehungsweise Dekom-

pressionsverfahren mehr Prozessorressourcen benötigen. Der SQL Server besitzt mittlerweile jedoch einen Abfrageausführungsmechanismus, mit dessen Hilfe die CPU-Nutzung bei der Anwendung von Column Stores minimiert werden kann.

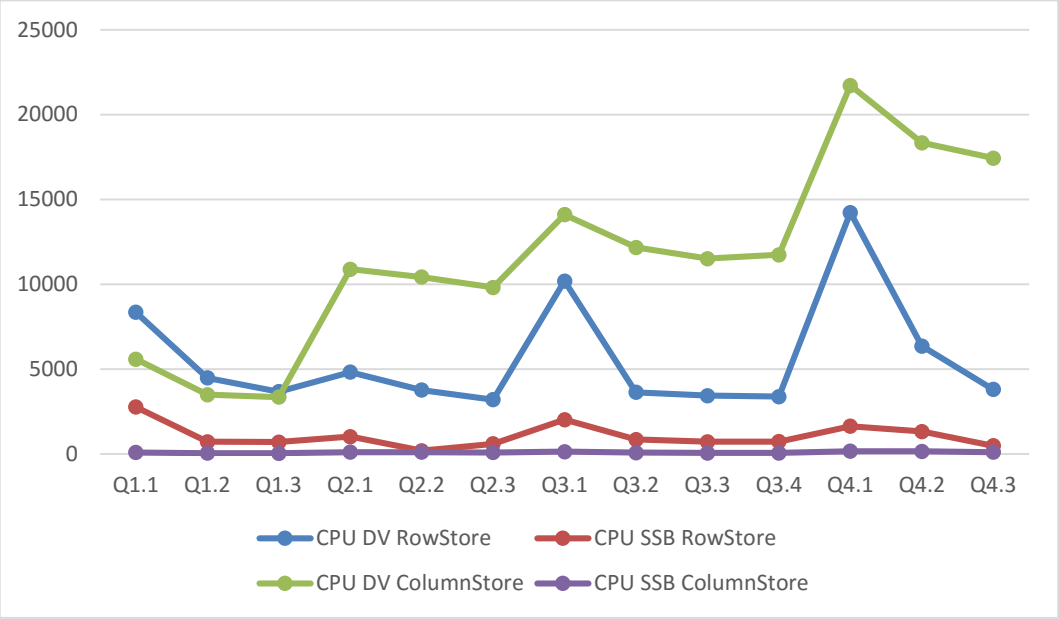


Abbildung 20: CPU Time der Abfragen basierend auf den verschiedenen Datenmodellen

Auch bei der Performance von Insert-und Delete-Abfragen gibt es große Unterschiede zwischen den beiden Modellen. Die Ursachen dafür ist die schon bereits erwähnte Aufteilung der Business Keys und beschreibenden Elemente von Business Objekten sowie die Modellierung deren Beziehungen. Abbildung illustriert diese Unterschiede.

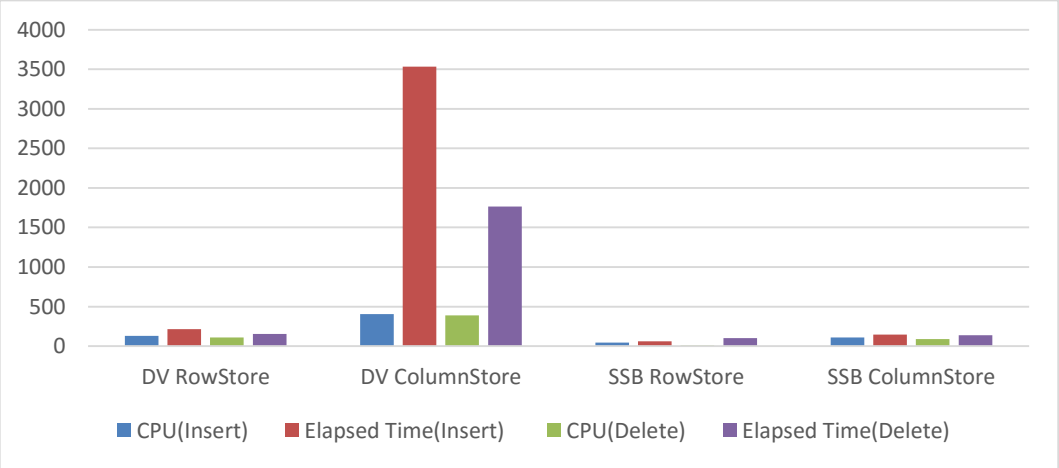


Abbildung 21: Laufzeit und Prozessorzeit der Aktualisierungsprozesse

Abschließend zum Vergleich beider Modelle hinsichtlich ihrer Performance kann festgestellt werden, dass das Starschema dem Data Vault in Sachen Laufzeit und Prozessorzeit wohl einiges voraushat, auch wenn die Laufzeiten beider Modelle sich in diesem Vergleich aufgrund der mini-

malistischen Größe des Data Vaults nur gering unterscheiden. Alle in den Diagrammen illustrierten Messwerte können den Tabellen 6 – 8 der Anhänge B.1 und B.2 entnommen werden.

Die These, dass die Laufzeitunterschiede bei der Wahl größerer Data Vault Modelle gravierender ausfallen, wird im Folgenden durch ein weiteres Beispiel unterstützt. Dabei wird sich ausgehend von dem soeben zum Vergleich benutzten Data Vault dem Erstentwurf aus Abbildung 17 angenähert. Zu den bereits vorhandenen Tabellen kommen mit der Tabelle „Nation“ und der Tabelle „Region“ zwei weitere hinzu, dessen Attribute sich in den Abfragen vom SSB häufig wiederfinden. Abbildung 22 zeigt einen Ausschnitt vom Data Vault Modell mit den neu hinzugekommenen Objekten Nation und Region. Auf dem Bild ist nicht zu erkennen, dass die Attribute „C_Nation“ und „S_Nation“ aus den Satelliten „Customer“ und „Supplier“ gelöscht wurden. Das vollständige Modell ist der Abbildung 25 im Anhang C.1 zu entnehmen.

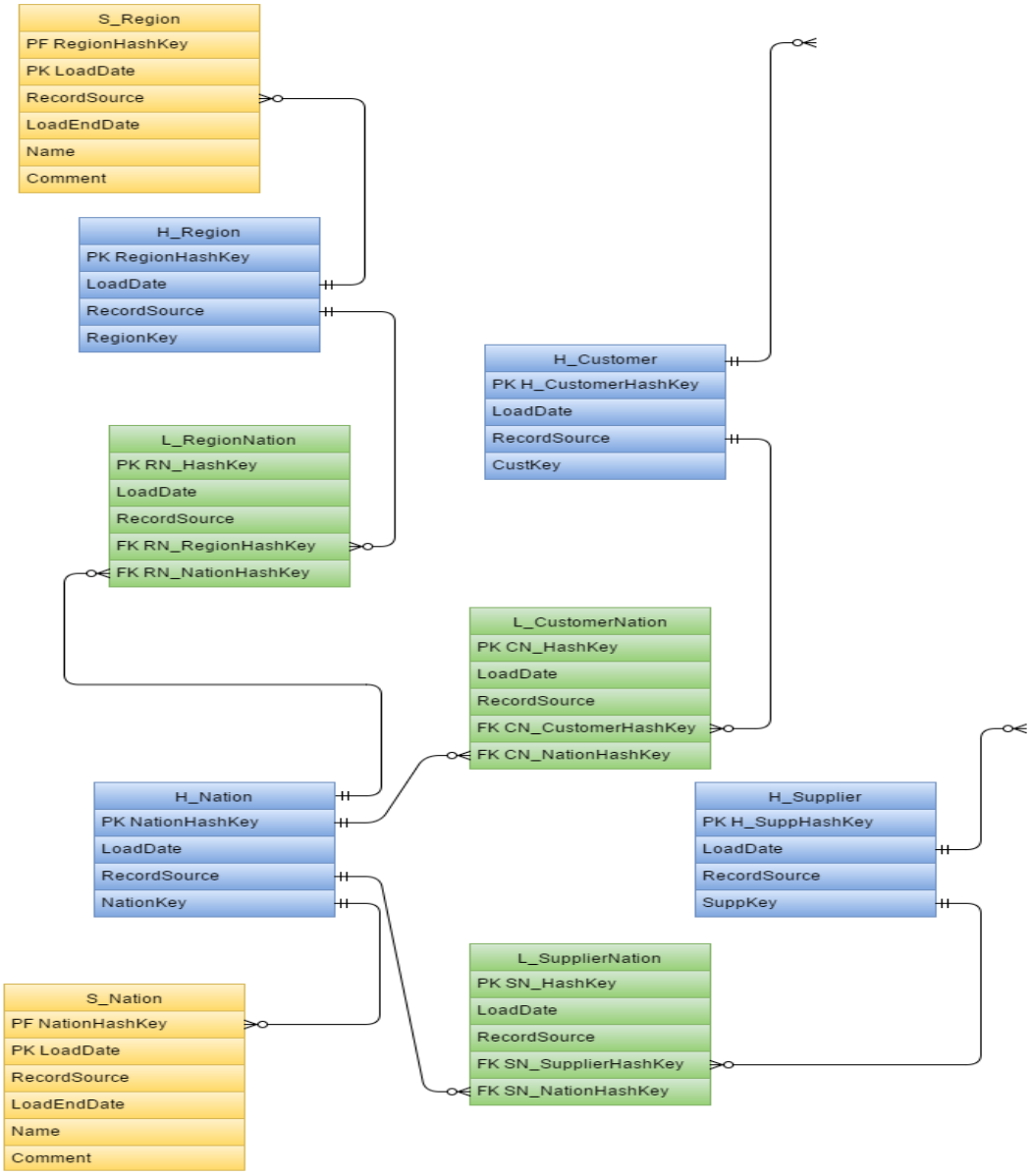


Abbildung 22: Ausschnitt modifiziertes Data Vault

Da dieses modifizierte Data Vault nun zwei neue Hubs, drei Links und zwei Satelliten mit Attributen besitzt, welche die Nation beziehungsweise Region charakterisieren, müssen die Abfragen Q2.1 – Q2.3, Q3.1, Q3.2 und Q4.1 – Q4.3 angepasst werden, da innerhalb dieser Abfragen deren Attribute zu finden sind. Zum Vergleich wird nun beispielhaft die bearbeitete Abfrage Q4.1 illustriert. Alle weiteren modifizierten Abfragen sind dem Anhang C.2 zu entnehmen:

```
SELECT d_year, t3.N_Name, sum(U_Revenue -lo_supplycost) as profit
FROM S_Umsatz,S_LineOrder, S_Date, S_Part, S_NATION t3,S_Nation t2, H_Supplier,
L_LineOrder, H_Date, H_Part, Link_nationsupplier,H_Nation n1, H_Nation n2, link_nationregion
ns1, link_nationregion ns2, H_region r1,H_Region r2, S_Region x1,S_Region x2, H_Customer,
link_nationcustomer
WHERE S_Umsatz.L_LOHashKey= L_LineOrder.L_LOHashKey
AND S_LineOrder.LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_DateHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND L_LineOrder.L_CustHashKey = H_Customer.H_CustHashKey
AND L_LineOrder.L_PartHashKey = H_Part.H_PartHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Part.H_PartHashKey = S_Part.PartHashKey
AND H_Supplier.H_SuppHashKey = link_nationsupplier.supplierhashkey
AND link_nationsupplier.nationhashkey = n1.n_nationhashkey
AND n1.n_nationhashkey = ns1.nationhashkey
AND ns1.regionhashkey = r1.H_Regionhashkey
AND r1.H_Regionhashkey = x1.R_REgionhashkey
AND H_Customer.H_CustHashKey = link_nationcustomer.customerhashkey
AND link_nationcustomer.nationhashkey = n2.n_nationhashkey
AND n2.n_nationhashkey = ns2.nationhashkey
AND n2.N_NATIONHashKey = t3.N_NATIONHashKey
AND ns2.regionhashkey  = r2.H_Regionhashkey
AND r2.H_Regionhashkey = x2.R_REgionhashkey
AND x1.R_Name = 'AMERICA'
AND x2.R_Name = 'AMERICA'
AND (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
GROUP BY d_year, t3.N_Name ORDER BY d_year, t3.N_Name;
```

Im Vergleich zur Abfrage Q4.1 ist klar zu erkennen, dass durch die neu hinzugekommenen Komponenten die Zahl der Verknüpfungen rapide ansteigt. Wie sich die Modifizierung des Data Vault Modells auf die Laufzeit ausgewirkt hat, ist der Abbildung 23 zu entnehmen. Die einzelnen Messwerte sind im Anhang C.3 in der Tabelle 9 zu finden.

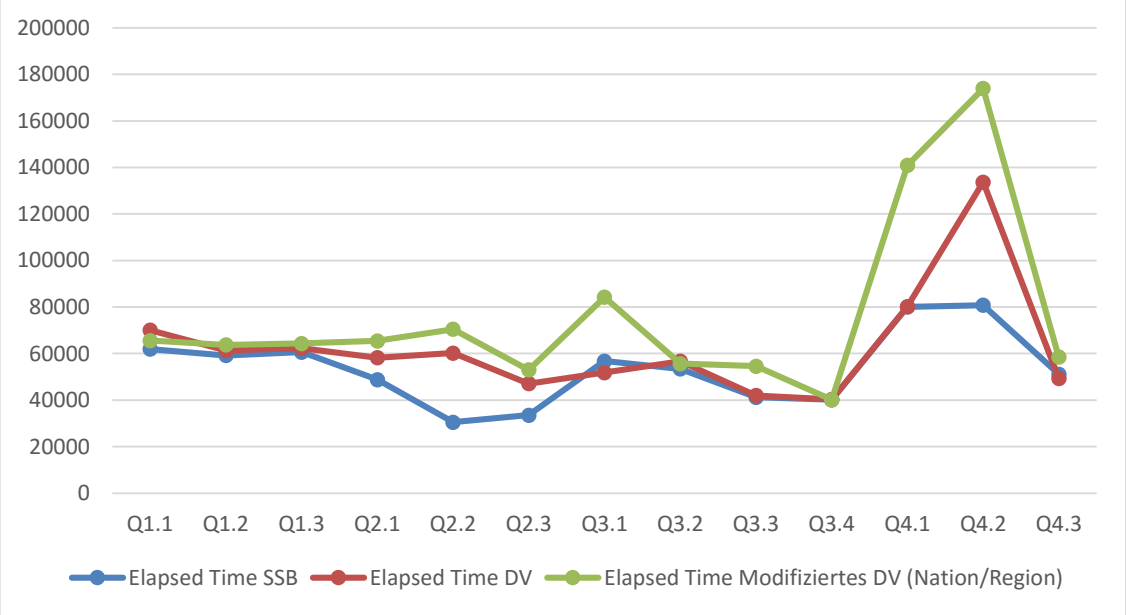


Abbildung 23: Laufzeit-Vergleich von modifizierten DV mit SSB und einfachen DV

In dieser Abbildung sind die Laufzeiten der einzelnen Abfragen bezogen auf die Datenmodelle dargestellt. Es ist deutlich erkennbar, dass die Abfragen, welche sich auf die Selektion oder Restriktion von Nation oder Region beziehen, beim größeren Data Vault eine niedrigere Performance aufweisen als beim alten.

Die Tatsache, dass Data Vault Modelle flexibel gestaltet und, wie eben am zweiten Modell illustriert, ständig weiterentwickelt werden, unterstützt die oben genannte These, dass die Performance eines Starschemas höher ist als die eines Data Vaults. Das minimalistische Modell, das hier zum Vergleich genutzt wurde, wird gar nicht oder kaum im Alltag beim Arbeiten mit Data Vault Systemen vorzufinden sein, da dies mit dem Ziel der Flexibilität in Widerspruch steht. Anhand dieses ersten Vergleichs der Modelle kann man erahnen, dass diese Flexibilität mit Einbußen in der Performance einhergeht, ohne jedoch spezielle Maßnahmen zur Performancesteigerung, wie zum Beispiel das Erstellen und Nutzen von materialisierten Sichten, bedacht zu haben.

8.7 Abfrage-Optimierung

Es gibt diverse Möglichkeiten, Optimierungen an Abfragen und deren zugrundeliegenden Modellen vorzunehmen. Eine übliche Methode zur Abfrage-Optimierung ist die Erstellung von Sichten. Sichten sind logische Relationen, die als im Datenbanksystem gespeicherte Abfragen den Zugriff

auf das Datenbankschema vereinfachen. Neben den Sichten gibt es für die Data Vault Modellierung zwei spezielle Methoden, um solch einen vereinfachten Zugriff auf das Schema zu gewährleisten. In den folgenden zwei Abschnitten werden die beiden Methoden näher erläutert und teilweise auf das Data Vault Modell angewendet.

8.7.1 Point-In-Time Tabellen

Als Point-in-Time (PIT) Tabellen werden modifizierte Satelliten beschrieben, die eingesetzt werden, um Abfragen, die auf mehreren Satelliten eines Hubs laufen, zu vereinfachen. Dafür werden in einer PIT-Tabelle alle Load Dates, also das Attribut, das den Zeitpunkt des Ladens in das Data Vault beschreibt, der einzelnen Datensätze von den Satelliten aufgeführt. Dies hat zum Vorteil, dass so Abfragen mit zeitlichem Aspekt effektiver umgesetzt werden können.
Besteht ein Hub aus mehreren Satelliten, sind meist die Ursachen unterschiedliche Datenquellen oder unterschiedliche Aktualisierungswahrscheinlichkeiten. Folglich werden Daten in Satelliten unterschiedlich aktualisiert. Baut man nun zum Beispiel ein Data Mart und fragt daher den kompletten Status eines Objekts an einem bestimmten Tag an, ist es teilweise schwierig und langsam, die zu diesem Objekt dazugehörenden Attribute der verschiedenen Satelliten zu bekommen, die an diesem Tag aktiv waren. Man müsste auf Outer Joins über den Hub zurückgreifen und komplexe Zeitspannen handhaben, was bei einem Hub mit vielen Satelliten schnell zu diesen Problemen führt. [Linstedt et al. 2015]
Um die Mengen an Datensätzen handhaben zu können und keine Performance-Probleme zu bekommen, wird eine PIT-Tabelle mit einem Snapshot-Datum versehen, also dem Datum, an dem die Datensätze in die PIT-Tabelle geschrieben wurden. Mit Hilfe des Snapshots und vorher festgelegten Regeln können dann Zeitfenster definiert werden, wann ein Ladevorgang in eine PIT-Tabelle erfolgt bzw. ob oder wann ältere Datensätze aus der PIT-Tabelle gelöscht werden sollen. Da dies eine vom System generierte Tabelle ist, ist diese auch nur im Business Vault vorhanden und kann nicht im Raw Vault modelliert werden. [Linstedt et al. 2015]
Aufgrund der Tatsache, dass in dem hier verwendeten Data Vault keine Hubs mit mehr als einem Satelliten existieren, kann die PIT-Tabelle nicht angewendet werden.

8.7.2 Bridge Tabellen

Während PIT-Tabellen auf Hubs mit mehreren Satelliten eingesetzt werden, spannt sich eine Bridge-Tabelle über mehrere Hubs und Links. Sie dient sozusagen als „Brücke“ und stellt eine direkte Verbindung zwischen Hubs her, die nur über mehrere Joins miteinander verknüpft werden können. Dadurch können Performance-Vorteile erzielt werden. Um das zu untermalen, wurde in

das erweiterte Data Vault Modell aus Abbildung 22 solch eine Bridge-Tabelle modelliert. Diese Bridge-Tabelle wurde über die Hubs „Nation“,„Region“ und „Supplier“ sowie den Links „Link_NationRegion“ und „Link_NationSupplier“ gespannt. Somit bestand die Bridge-Tabelle „Renasu“ aus folgenden Attributen:

Tabelle 5: Beispiel Bridge-Tabelle „Renasu“

Region-HashKey	NR_HashKey	NationHash-Key	NS_HashKey	Supplier-HashKey	Snapshot
0d85516…	151a41sa4s...	0787b38aeea…	8485sas15…	0042d8…	2106-04-03
0d85516…	151a41sa4s...	0787b38aeea…	454s8asa…	Gg1s51…	2106-04-03
…	…	…	…	…	…

Neben den Hash-Schlüsseln aller miteinander verknüpften Tabellen wird auch ein Snapshot mit in die Tabelle eingefügt, der die Zeit des Ladens in die Bridge-Tabelle festhält.
Bevor der Power-Test mit neu hinzugewonnener Bridge-Tabelle absolviert werden konnte, mussten die Anfragen Q2.1 – Q2.3, Q3.1, Q3.2 und Q4.1 – Q4.3 einmal mehr optimiert werden. Q4.1 wird wieder zu Verdeutlichung herangezogen:

```
SELECT d_year, N_Name, sum(U_Revenue - lo_supplycost) as profit
FROM S_Umsatz,S_LineOrder,bridge_renasu, S_Date, S_Part, S_NATION, H_Supplier,
L_LineOrder, H_Date, H_Part, H_Nation, link_nationregion, H_region r1,H_Region r2, S_Region
x1,S_Region x2, H_Customer, link_nationcustomer
WHERE S_Umsatz.L_LOHashKey= L_LineOrder.L_LOHashKey
AND S_LineOrder.LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_DateHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND L_LineOrder.L_CustHashKey = H_Customer.H_CustHashKey
AND L_LineOrder.L_PartHashKey = H_Part.H_PartHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Part.H_PartHashKey = S_Part.PartHashKey
AND H_Supplier.H_SuppHashKey = bridge_renasu.supplier_hashkey
AND bridge_renasu.region_hashkey = r1.h_regionhashkey
AND r1.H_Regionhashkey = x1.R_REgionhashkey
```

AND H_Customer.H_CustHashKey = link_nationcustomer.customerhashkey
AND link_nationcustomer.nationhashkey = H_Nation.n_nationhashkey
AND H_Nation.n_nationhashkey = link_nationregion.nationhashkey
AND H_Nation.N_NATIONHashKey = S_Nation.N_NATIONHashKey
AND link_nationregion.regionhashkey = r2.H_Regionhashkey
AND r2.H_Regionhashkey = x2.R_REgionhashkey
AND x1.R_Name = 'AMERICA'
AND x2.R_Name = 'AMERICA'
AND (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
GROUP BY d_year, N_Name ORDER BY d_year, N_Name;

Durch das Zusammenfügen der Primärschlüssel von Region, Nation und Supplier in einer Tabelle war es nun möglich, mit weniger Joins das gleiche Ergebnis zu erzielen. Alle modifizierten Abfragen sind unter Anhang D.1 zu finden.

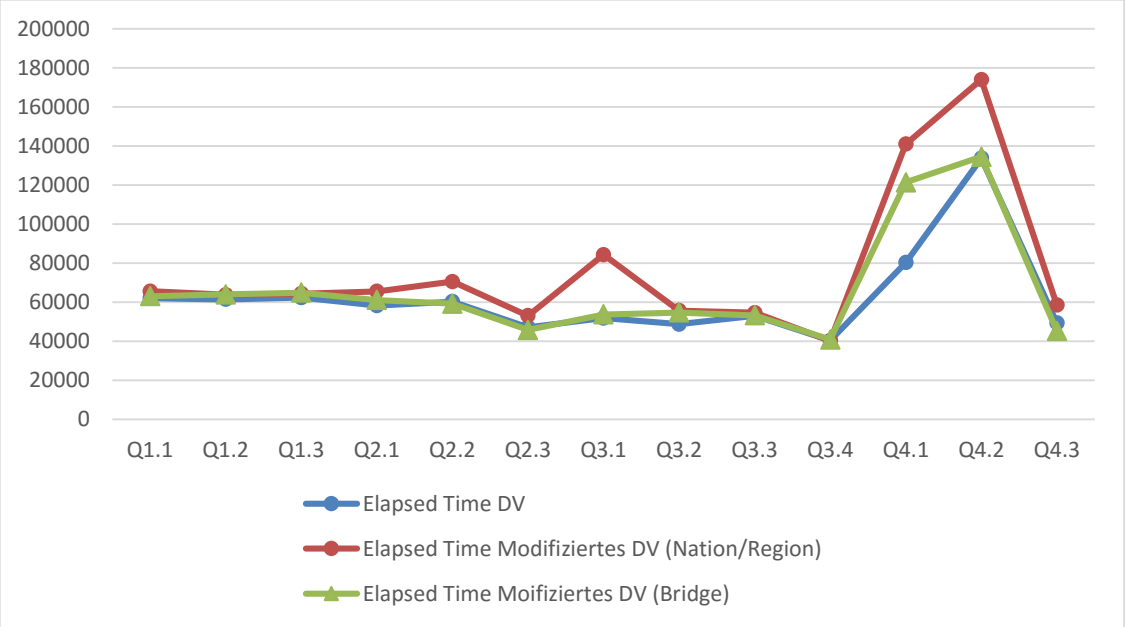


Abbildung 24: Laufzeit-Vergleich von DV-Bridge Table, DV-modifiziert, DV-normal

Abbildung 24 zeigt die Ergebnisse vom Power-Test mit Bridge-Tabelle verglichen mit dem Power-Test vom minimalistischen Data Vault und modifiziertem Data Vault. Es ist gut zu sehen, dass durch das Einfügen einer Bridge Tabelle die Laufzeit gegenüber dem in Abschnitt 8.6 modfiziertem Data Vault gesunken ist. Vor allem bei den Abfragen Q3.1, Q4.1 und Q4.2 lassen sich Performance-Unterschiede erkennen. Die einzelnen Messwerte können der Tabelle 10 im Anhang D.2 entnommen werden. Der Power-Test mit Bridge-Tabelle hat aufgezeigt, dass dies eine Möglichkeit zur Optimierung der Performance bietet, wenn Abfragen über eine Vielzahl an Hubs durch

indirekte Joins die Alternative sind. Nichtdestotrotz sind die Bridge-Tabellen mit Vorsicht zu genießen und nur bei schwerwiegenden Performanceproblemen in Betracht zu ziehen, da sie neben der in Inanspruchnahme des physischen Speichers auch das Data Vault Modell komplexer werden lassen. [Linstedt et al. 2015]

8.8 Bewertung der Messung

Im Hinblick auf Unterschiede bei der Performance zwischen beiden Datenmodellen liefert der in Abschnitt 8 absolvierte quantitative Vergleich keine komplett aussagekräftigen Ergebnisse, da nur mit einem konstanten Datenbestand gearbeitet wurde und somit verschieden große Datenbestände außen vorgelassen wurden. Weiterhin wurde zum Vergleich nicht das Raw Vault, sondern ein Business Vault herangezogen, welches durch voraggregierte Attribute bzw. vom System generierte Tabellen schon voroptimiert war. Dennoch können die in Abschnitt 5 gezogenen ersten Erkenntnisse über das Performance-Verhalten beider Datenmodelle bei der Abfragebearbeitung in dieser Messung bestätigt werden. Das Data Vault Modell hat gegenüber dem Starschema höhere Laufzeiten bei fast allen Abfragen, auch wenn diese Unterschiede eher gering ausfallen. Der spaltenorientierte Ansatz durch die Columnstores bringt bei der Abfragebearbeitung enorme Laufzeitverbesserungen mit sich, ist jedoch weniger als Speichervariante für das Core-Data Warehouse, in das bei der Data Vault Architektur die Rohdaten geladen und historisiert werden, aufgrund der zeitaufwendigen Aktualisierungsprozesse geeignet.

Durch diverse Optimierungsmöglichkeiten der Abfragebearbeitung können Laufzeiten minimiert werden. Die beiden hier vorgestellten Methoden der PIT- und Bridge-Tabellen zeigen auf, dass auch im Data Vault bezüglich der Bildung von Data Marts Optimierungsmöglichkeiten bestehen und unterschiedliche Ursachen bei Performance-Einbußen behandelt werden können. Inwiefern sich das Konzept der Sichten auf das Data Vault Modell auswirkt, bleibt einem dieser quantitative Vergleich schuldig und muss in weiteren Betrachtungen analysiert werden.

9. Zusammenfassung

Mit Data Vault 2.0 ist eine neue komplexe architektonische Erweiterung für Data Warehouses entstanden, die nicht nur mit einer Referenzarchitektur, sondern auch einen neuen Ansatz eines Datenmodells, eine agile Methodologie und Best Practices für die Implementation beinhaltet. Neben dem Konzept paralleler Architekturen, schließt Data Vault die Thematiken rund um Big Data, Echtzeit-Laden und der Integration unstrukturierter Daten mit ein.

Einer der Unterschiede zu traditionellen Data Warehouses besteht in der drei-schichtigen Referenzarchitektur, durch welche die Business Regeln zum Modifizieren von Daten in Richtung Endnutzer geschoben werden können und somit der Workload beim Extrahieren von Daten auf den operativen Systemen reduziert wird. Die Staging Area als erste Schicht extrahiert dementsprechend lediglich die Daten aus den Datenquellen und lädt diese in eine separate Datenbank, dem Data Vault. Da auch unstrukturierte Datenbestände schnellstmöglich extrahiert werden müssen, sollte eine Data Vault Architektur als Staging Area auch NOSQL-Plattformen wie Hadoop unterstützen.

Im Kern des Data Warehouses, dem Edw-Layer, können die durch die Staging Area extrahierten Daten dann historisch aufbereitet und dank den durch das Datenmodell beschriebenen Komponenten bestehend aus Hubs, Links und Satelliten flexibel umgesetzt werden.

Durch diese Flexibilität bei Erweiterungen ist die Projektion anderer Data Warehouse Modelle auf ein Data Vault Modell gewährleistet und einfach zu realisieren, wie man am Beispiel des SSB-Schemas in Abschnitt 8.1 und 8.2 erkennen kann.

Das Ziel der Masterarbeit, das Data Vault Modell mit den traditionellen multidimensionalen Speichervarianten zu vergleichen, wurde sowohl durch eine qualitative Analyse anhand von Data Warehouse Anforderungen als auch einen quantitativen Vergleich anhand von Performance-Messungen durchgeführt. Diese Vergleiche geben erste Hinweise auf die Stärken und Schwächen des jeweiligen Modells.

Aus der Sicht der Endnutzer eines Data Warehouses besticht vor allem das Starschema durch Performance-Vorteile und der Einfachheit der Struktur des Datenmodells und der Abfragekomplexität, während das Data Vault Modell durch die Flexibilität bei Erweiterungen wesentliche Vorteile aufweisen kann. Unter anderem können Bridge- und PIT Tabellen zu Steigerungen der Performance der Abfragebearbeitung führen. Nichtsdestotrotz geht mit dieser flexiblen Handhabung die Übersichtlichkeit eines solchen Modells schnell verloren. Das neutrale Design der Links hat außerdem den Nachteil, dass man bei großen Modellen den Business-Kontext nicht mehr aus dem Modell „ablesen“ kann und damit die Zusammenhänge nicht mehr automatisch ersichtlich

sind. Vom Standpunkt der Abfragebearbeitung ausgesehen, ist das Data Vault außerdem ziemlich komplex, was sich anhand des quantitativen Vergleichs mit dem Starschema beweisen lässt. Die Verteilung der Daten auf mehrere Hubs, Satelliten und Links führt beim Erstellen von Reports direkt im Data Vault zu sehr komplexen SQL-Statements, in welchen viele Tabellen miteinander verknüpft werden müssen. Dies führt oft unweigerlich zu schlechten Performance-Ergebnissen.
Um die Probleme der Performance zu umgehen, existiert in der Data Vault Architektur mit dem Information Mart Layer eine Schicht, in der aus dem Data Vault abgeleitete Data Marts mit besser geeigneten Strukturen für Reporting-Analysen, wie das Starschema, umgesetzt werden. Da die Data Marts ihre Daten aus dem Data Vault beziehen, müssen diese periodisch mit Hilfe von ETL-Prozessen aktualisiert werden, um deren Datenbestand zeitaktuell zu halten. Diese ETL-Prozesse erfordern einen zeit- und kostenintensive Erstellungs- und Instandhaltungsaufwand. Werden Änderungen im Data Vault Modell vorgenommen, indem zum Beispiel ein neuer Satellit eingefügt wird, müssen alle ETL-Prozesse, die auf diese Datenstrukturen zugreifen, überarbeitet werden. Weiterhin ist davon auszugehen, dass viele Data Marts auf gleiche Datenstrukturen zugreifen und somit Daten redundant gespeichert werden.
Zusammenfassend kann man sagen, dass die Flexibilität beim Integrieren neuer Datenbestände aus unterschiedlichen Quellen durch die aus den schlechten Performance-Ergebnissen resultierende notwendige Erstellung von Data Marts eingeschränkt wird. Die Data Vault Architektur bietet dennoch einen sehr guten Ansatz für agile Data Warehouses, da mit dem Data Vault Modell ein sehr flexibles Datenmodell entwickelt wurde, welches sich bei Veränderungen der Unternehmenslogik beliebig anpassen lässt. Allerdings hat es in Bezug auf die Reporting-Funktionen im Data Warehouse Nachteile gegenüber dem Starschema. Ein Zusammenspiel beider Varianten in einer Architektur, wie in Data Vault 2.0, klingt daher plausibel. Auch wenn das Starschema bereits in der Data Vault Architektur eingesetzt wird, um die für den Endnutzer wichtigen Daten bereitzustellen, sollte jedoch anhand der oben genannten Nachteile bei der Umsetzung eines Data Warehouses auf Basis der Data Vault Architektur vor allem der Information Mart Layer überdacht und Ansätze für schnellere und einfachere Reporting-Funktionen entwickelt werden. Rick van der Lans, ein unabhängiger BI-Analyst aus den Niederlanden, hat kürzlich bezüglich dieser Probleme auf Basis von Daten-Virtualisierung ein Konzept mit dem Namen „SuperNova“ vorgestellt. [Rick F. van der Lans 2015]

10. Ausblick

Aufgrund der beschränkten Möglichkeiten bezüglich der Hard- und auch Software konnten keine Vergleiche beider Datenmodelle auf der Basis großer Datenbanken stattfinden. Um die in dieser Arbeit erlangten quantitativen Ergebnisse zu bestätigen, sind weitere Messungen auf verschiedenen Datenbankgrößen zu absolvieren. Dabei wäre es von Vorteil, sich nicht nur auf das kleine Data Vault, das aus dem SSB gewonnen wurde, zu beschränken, sondern auch das aus dem TPC-H Schema abgeleitete komplexe Data Vault in die Vergleiche mit einzubeziehen. Auf dessen Grundlage können somit auch die in Abschnitt 8.5 beschriebenen Verfahren der PIT- und Bridge-Tabellen zur Optimierung der Abfragebearbeitung ausführlicher angewendet und deren Auswirkungen auf die Performance untersucht werden. Des Weiteren kann das Konzept der materialisierten Sichten als weitere Optimierungsmethode betrachtet werden, um auf dessen Basis vergleichende Analysen stattfinden zulassen, damit ausführliche und allgemeingültige quantitativer Ergebnisse bezüglich der Datenmodelle präsentiert werden können.

In Bezug auf die komplette Data Vault Architektur gibt es vor allem beim Information Mart Layer Verbesserungsbedarf. Da sowohl die Staging Area als auch der EDW-Layer durch ihre Eigenschaften in der Parallelisierung von Ladeprozessen und Flexibilität überzeugen und die erstrebenswerte Agilität eines Data Warehouses erhöhen, können weitere Ansätze darauf aufbauen, Konzepte zur Optimierung der Reporting-Performance im Information Mart, wie zum Beispiel van der Lans‘ SuperNova [Rick F. van der Lans 2015], anzuwenden und mit den allgemein üblichen Konzepten des Data Vaults qualitativ als auch quantitativ zu vergleichen.

Mit Hilfe dieser Analysen sowohl auf Datenmodell- als auch Architekturebene können weitere Bewertungen hinsichtlich der Notwendigkeit des Information Mart Layers und diesbezüglich die des Starschemas in der Data Vault Architektur getroffen werden.

Anhang A: TPC-H Abfragedefinitionen

Aufgelistet sind alle Abfragen des TPC-H Benchmarks. Diese wurden aus [TPC 2014] umformuliert:

Q1 Preisbericht: Es wird ein Preisbericht erstellt in dem alle Auftragspositionen in einem bestimmten Zeitraum verzeichnet sind. Diese sollen nach den Attributen LINESTATUS und RETURNFLAG gruppiert werden. Für jede Gruppe sollen umsatzrelevante Daten ausgegeben werden, beispielsweise der Gesamtpreis oder die Anzahl.

Q2 Günstigster Lieferant: Es wird ermittelt welcher Zulieferer für ein bestimmtes Teil gewählt werden sollte. Kriterien sind dabei der Standort des Lieferanten und die Lieferkosten. Generiert wird ein Bericht, der alle qualifizierten Lieferanten mit Kontakt- und Umsatzdaten enthält.

Q3 Versand Priorität: Es wird ausgeben welche Aufträge aus einem bestimmten Marktsegment noch nicht ausgeliefert wurden. Kriterien sind, dass nur die 10 Aufträge mit dem höchsten Umsatz ausgegeben werden, wobei der Fokus auf der Lieferpriorität liegt.

Q4 Bestellpriorität Verifizierung: Hierbei wird festgestellt wie gut das System zur Festlegung der Lieferpriorität funktioniert. Dafür werden alle Datensätze ausgegeben, die in einem bestimmten Zeitraum bestellt wurden, aber nach dem zugesagten Lieferdatum ankamen.

Q5 Lokales Händler Volumen: Es soll festgestellt werden ob es sich lohnt in einer bestimmten Region Verteilungszentren zu errichten. Basis für diese Erhebung sind die Umsatzdaten einer Region. Es werden nur Händler und Kunden aus derselben Region untersucht.

Q6 Ertragsvorhersage: Es wird untersucht um welche Summe sich der Ertrag in einem bestimmten Jahr erhöht hätte, falls ein Rabatt für eine bestimmte Menge an Teilen nicht gewährt worden wäre.

Q7 Transportvolumen: Die Abfrage gibt an welcher Warenwert in einem Zeitraum zwischen zwei Nationen verschickt wurde. Ziel ist die Überarbeitung der Lieferverträge.

Q8 Marktanteil: Es wird errechnet welchen Marktanteil eine Nation in einer Region bei einem gegebenen Produkt in den letzten zwei Jahren hat.

Q9 Ertrag: Es wird ermittelt mit welcher Produktlinie in einem Zeitraum der größte Gewinn erwirtschaftet werden konnte.

Q10 Rückgabe: Es werden Kundendaten ausgegeben, die die Kontaktdaten enthalten. Das Kriterium ist hierbei der Umsatzverlust, der durch Rücksendungen entstanden ist. Es werden die Top 20 ausgegeben.

Q11 Lagerbestand: Es soll ermittelt werden welche Teile in einer bestimmten Region gelagert sind und einen signifikanten Anteil am Gesamtwarenwert haben. Der Gesamtwarenwert bezieht sich nur auf die in einer Nation lagernden Teile.

Q12 Versand Analyse: Es wird eine Aussage darüber getroffen, ob das Wechseln zu einer günstigeren Lieferart, kritische Aufträge negativ beeinflussen kann.

Q13 Umsatzverlust: Es wird festgestellt welcher Sachbearbeiter, in einem bestimmten Zeitraum, den höchsten Umsatzverlust durch Rücksendungen verursacht hat.

Q14 Werbung: Es wird ermittelt, ob durch die Schaltung von Werbung zu einem höheren Umsatz gekommen ist. Betrachtet werden hierbei nur die Produkte die auch beworben wurden.

Q15 Top Händler: Es wird der beste Lieferant ermittelt, wobei hier der Anteil am Gesamtumsatz ausschlaggebend ist.

Q16 Händlerlager: Es wird ermittelt welche Lieferanten eine bestimmte Sorte von Teilen auf Lager haben, wobei Lieferanten ausgeschlossen werden, die eine negative Bewertung bekommen haben.

Q17 Kleine Lieferungen: Es wird der Verlust berechnet, der durch die Ablehnung kleiner Bestellungen eines bestimmten Teils, entstehen würde.

Q18 Große Bestellungen: Es werden die Top 100 Kunden ausgegeben, die das größte Umsatzvolumen haben, wobei die Bestellungen eine Mindestanzahl an Einheiten haben müssen.

Q19 Gesamtumsatz: Es soll bestimmt werden, welches Produkt den höchsten Umsatz erzielt hat, wobei entscheidend ist wie das Produkt ausgeliefert wurde.

Q20 Werbe Aktion: Es wird ermittelt, ob es in einer bestimmten Nation Lagerbestände existieren, die sich für eine Rabattaktion eigenen.

Q21 Versandverzögerung: Die Abfrage gibt die Händler wieder, die mit der Auslieferung von Teilen in Verzug geraten sind.

Q22 Geographische Nachfrage: Die Abfrage schlüsselt auf, ob Kunden existieren die wieder kontaktiert werden sollten. Die Aufschlüsselung erfolgt über den Gesamtumsatz des Kunden und den Zeitraum in dem von Ihm nichts bestellt wurde.

Anhang B: Messwerte der Analyse von SSB und Data Vault

B.1 Laufzeit- und CPU-Messwerte von SSB und Data Vault Abfragen

Tabelle 6: Laufzeitmesswerte von SSB und DV spaltenorientiert (CS) und zeilenorientiert

Abfrage	Elapsed Time SSB	Elapsed Time DV	Elapsed Time SSB CS	Elapsed Time DV CS
Q1.1	61982,591	70047,262	1393,756	4141,203
Q1.2	59208,621	61367,267	1297,133	5077,08
Q1.3	60653,121	62320,146	1322,194	5027,703
Q2.1	48813,863	58213,626	1980,167	8591,389
Q2.2	30534,933	60260,707	2002,79	8966,786
Q2.3	33560,03	47108,313	1980,308	8212,861
Q3.1	56801,662	51817,587	1968,599	9642,607
Q3.2	53411,981	56721,418	2103,969	9269,98
Q3.3	41228,144	42017,481	1975,676	8555,055
Q3.4	40203,625	40310,107	1904,297	8556,544
Q4.1	80118,045	80213,251	2796,628	14795,662
Q4.2	80811,57	133709,941	2904,781	13728,316
Q4.3	51155,866	49349,53	2762,447	13615,949

Tabelle 7: CPU-Messwerte von SSB und DV in spaltenorientiert (CS) und zeilenorientiert

Abfrage	CPU sek DVB RS	CPU sek SSB RS	CPU in ms DVB CS	CPU in ms SSB CS
Q1.1	8352,537	2775,481	5590,099	96,121
Q1.2	4490,837	730,523	3500,524	59,061
Q1.3	3680,279	709,023	3354,836	52,5
Q2.1	4825,49	1027,961	10894,236	111,295
Q2.2	3776,254	194,331	10429,739	110,667
Q2.3	3207,449	606,72	9817,881	91,093
Q3.1	10195,881	2025,699	14113,023	138,869
Q3.2	3639,571	859,059	12174,493	86,56
Q3.3	3443,205	728,228	11517,666	66,989
Q3.4	3390,809	732,501	11740,644	64,737
Q4.1	14227,338	1641,328	21724,578	171,304
Q4.2	6364,041	1334,547	18343,033	161,706
Q4.3	3808,809	487,514	17443,044	113,212

B.2 Messwerte der Aktualisierungsprozesse

Tabelle 8: Messwerte der Aktualisierungsprozesse Insert und Delete

Datenmodell	Insert CPU	Insert Elapsed Time	Delete CPU	Delete
DV RS	129,627	216,372	110,198	152,738
DV CS	407,037	3532,78	389,39	1763,58
SSB RS	47,26	61,939	8,965	102,965
SSB CS	111,47	148,47	90,608	139,729

Anhang C: Data Vault mit den Objekten Nation und Region

C.1 Abbildung des Modells

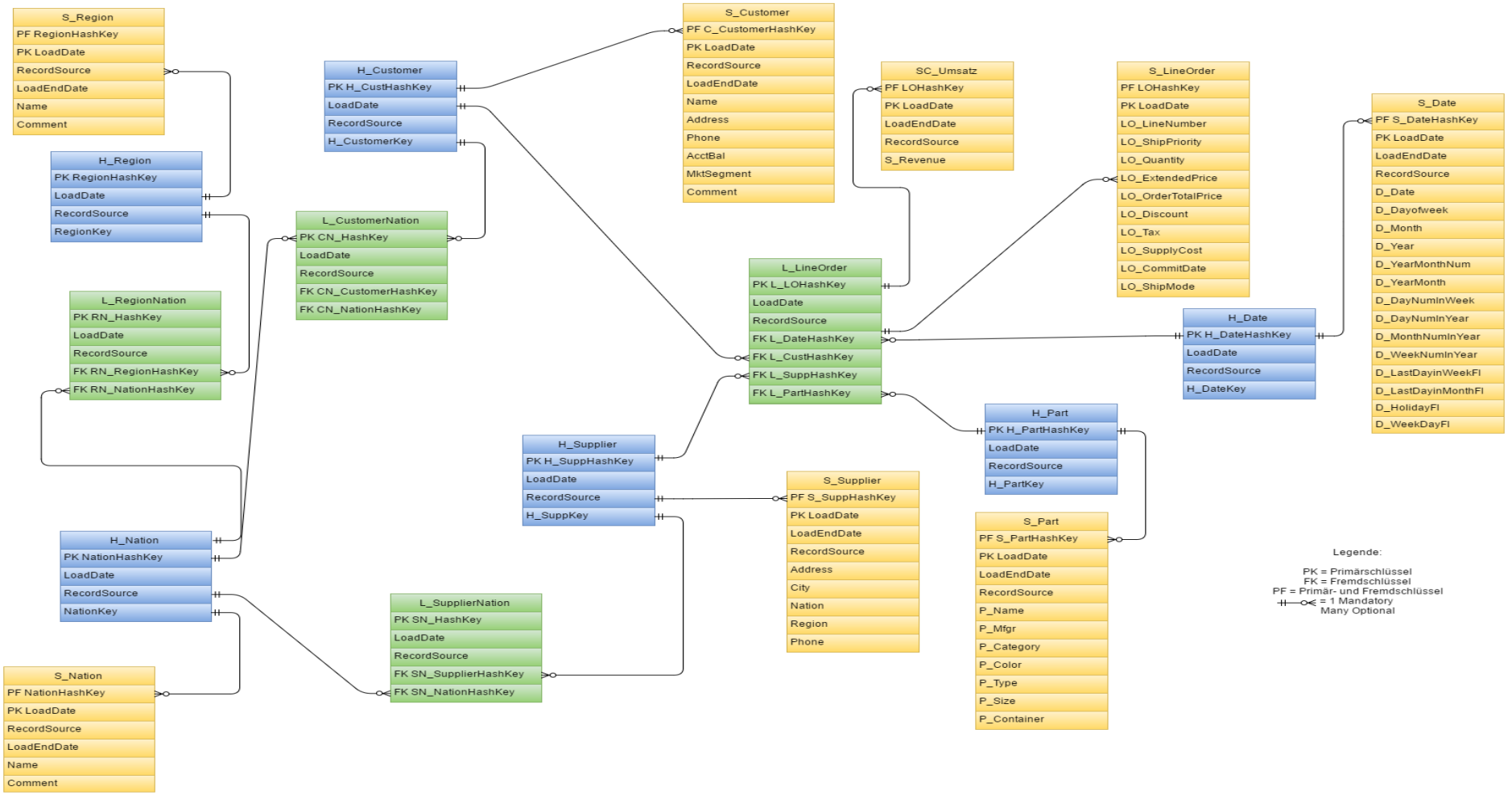


Abbildung 25: Erweitertes Data Vault Modell um die Objekte Nation und Region

C.2 Abfragen des erweiterten Data Vaults

Q2.1

```
SELECT sum(U_Revenue), D_year, P_brand
FROM S_Umsatz, S_Date, S_Part, H_Supplier, L_LineOrder, H_Date, H_Part,
Link_nationsupplier,H_Nation, link_nationregion, H_region, S_Region
WHERE S_Umsatz.L_LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_DateHashKey
AND L_LineOrder.L_PartHashKey = H_Part.H_PartHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Part.H_PartHashKey = S_Part.PartHashKey
AND H_Supplier.H_SuppHashKey = link_nationsupplier.supplierhashkey
AND link_nationsupplier.nationhashkey = H_Nation.n_nationhashkey
AND H_Nation.n_nationhashkey = link_nationregion.nationhashkey
AND link_nationregion.regionhashkey = H_Region.H_Regionhashkey
AND H_Region.H_Regionhashkey = S_Region.R_REgionhashkey
AND P_category = 'MFGR#12'
AND R_Name = 'AMERICA'
GROUP BY D_Year, P_Brand
ORDER BY D_year, P_brand;
```

Q2.2

```
SELECT sum(U_Revenue), D_year, P_brand
FROM S_Umsatz, S_Date, S_Part, H_Supplier, L_LineOrder, H_Date, H_Part,
Link_nationsupplier,H_Nation, link_nationregion, H_region, S_Region
WHERE S_Umsatz.L_LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_DateHashKey
AND L_LineOrder.L_PartHashKey = H_Part.H_PartHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Part.H_PartHashKey = S_Part.PartHashKey
AND H_Supplier.H_SuppHashKey = link_nationsupplier.supplierhashkey
AND link_nationsupplier.nationhashkey = H_Nation.n_nationhashkey
```

```
AND H_Nation.n_nationhashkey = link_nationregion.nationhashkey
AND link_nationregion.regionhashkey = H_Region.H_Regionhashkey
AND H_Region.H_Regionhashkey = S_Region.R_REgionhashkey
AND P_Brand between 'MFGR#2221' and 'MFGR#2228'
AND R_Name = 'ASIA'
GROUP BY D_Year, P_Brand
ORDER BY D_year, P_brand;
```

Q2.3

```
SELECT sum(U_revenue), D_year, P_brand
FROM S_Umsatz, S_Date, S_Part, H_Supplier, L_LineOrder, H_Date, H_Part,
Link_nationsupplier,H_Nation, link_nationregion, H_region, S_Region
WHERE S_Umsatz.L_LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_DateHashKey
AND L_LineOrder.L_PartHashKey = H_Part.H_PartHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Part.H_PartHashKey = S_Part.PartHashKey
AND H_Supplier.H_SuppHashKey = link_nationsupplier.supplierhashkey
AND link_nationsupplier.nationhashkey = H_Nation.n_nationhashkey
AND H_Nation.n_nationhashkey = link_nationregion.nationhashkey
AND link_nationregion.regionhashkey = H_Region.H_Regionhashkey
AND H_Region.H_Regionhashkey = S_Region.R_REgionhashkey
AND P_Brand = 'MFGR#2339'
AND R_Name = 'EUROPE'
GROUP BY D_Year, P_Brand
ORDER BY D_year, P_brand;
```

Q3.1

```
SELECT distinct t2.N_Name, t3.N_Name, D_Year, sum(U_Revenue) as revenue
FROM S_Umsatz, S_Date, S_NATION t3,S_Nation t2, H_Supplier, L_LineOrder, H_Date,
Link_nationsupplier,H_Nation n1, H_Nation n2, link_nationregion ns1, link_nationregion ns2,
H_region r1,H_Region r2, S_Region x1,S_Region x2, H_Customer, link_nationcustomer
where S_Umsatz.L_LOHashKey= L_LineOrder.L_LOHashKey
```

```
AND L_LineOrder.L_DateHashKey = H_DateHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND L_LineOrder.L_CustHashKey = H_Customer.H_CustHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Supplier.H_SuppHashKey = link_nationsupplier.supplierhashkey
AND link_nationsupplier.nationhashkey = n1.n_nationhashkey
AND n1.N_NATIONHashKey = t2.N_NATIONHashKey
AND n1.n_nationhashkey = ns1.nationhashkey
AND ns1.regionhashkey = r1.H_Regionhashkey
AND r1.H_Regionhashkey = x1.R_REgionhashkey
AND H_Customer.H_CustHashKey = link_nationcustomer.customerhashkey
AND link_nationcustomer.nationhashkey = n2.n_nationhashkey
AND n2.n_nationhashkey = ns2.nationhashkey
AND ns2.regionhashkey  = r2.H_Regionhashkey
AND r2.H_Regionhashkey = x2.R_REgionhashkey
AND n2.N_NATIONHashKey = t3.N_NATIONHashKey
AND x1.R_Name = 'ASIA'
AND x2.R_Name = 'ASIA'
AND d_year >= 1992 and d_year <= 1997
Group By t2.N_Name, t3.N_Name, D_Year
Order By D_Year asc, Revenue desc;
```

Q3.2

```
SELECT C_City, S_City, D_Year, sum(U_Revenue) as revenue
From S_Umsatz, S_Date, S_Supplier, L_LineOrder, H_Date, H_Supplier, S_Customer,
H_Customer, link_nationcustomer, link_nationsupplier, H_nation n1, H_Nation n2, S_Nation s1,
S_Nation s2
WHERE S_Umsatz.L_LOHashKey= L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_DateHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND L_LineOrder.L_CustHashKey = H_Customer.H_CustHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Supplier.H_SuppHashKey = S_Supplier.S_SuppHashKey
AND H_Supplier.H_SuppHashKey = link_nationsupplier.supplierhashkey
```

```
AND link_nationsupplier.nationhashkey = n1.N_NATIONHashKey
AND n1.N_NATIONHashKey = s1.N_NATIONHashKey
AND H_Customer.H_CustHashKey = S_Customer.S_CustHashKey
AND H_Customer.H_CustHashKey = link_nationcustomer.customerhashkey
AND link_nationcustomer.nationhashkey = n2.N_NATIONHashKey
AND n2.N_NATIONHashKey = s2.N_NATIONHashKey
AND s1.N_Name = 'UNITED STATES'
AND s2.N_Name = 'UNITED STATES'
AND d_year >= 1992 and d_year <= 1997
GROUP BY C_City, S_City, D_Year
ORDER BY D_Year asc, revenue desc;
```

Q4.2

```
SELECT d_year, t2.N_Name, p_category, sum(U_revenue -
lo_supplycost) as profit
from S_Umsatz,S_LineOrder, S_Date, S_Part,S_Nation t2, H_Supplier, L_LineOrder, H_Date,
H_Part, Link_nationsupplier,H_Nation n1, H_Nation n2, link_nationregion ns1, link_nationregion
ns2, H_region r1,H_Region r2, S_Region x1,S_Region x2, H_Customer, link_nationcustomer
where S_Umsatz.L_LOHashKey= L_LineOrder.L_LOHashKey
AND S_LineOrder.LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_DateHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND L_LineOrder.L_CustHashKey = H_Customer.H_CustHashKey
AND L_LineOrder.L_PartHashKey = H_Part.H_PartHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Part.H_PartHashKey = S_Part.PartHashKey
AND H_Supplier.H_SuppHashKey = link_nationsupplier.supplierhashkey
AND link_nationsupplier.nationhashkey = n1.n_nationhashkey
AND n1.N_NATIONHashKey = t2.N_NATIONHashKey
AND n1.n_nationhashkey = ns1.nationhashkey
AND ns1.regionhashkey = r1.H_Regionhashkey
AND r1.H_Regionhashkey = x1.R_REgionhashkey
AND H_Customer.H_CustHashKey = link_nationcustomer.customerhashkey
AND link_nationcustomer.nationhashkey = n2.n_nationhashkey
```

```
AND n2.n_nationhashkey = ns2.nationhashkey
AND ns2.regionhashkey  = r2.H_Regionhashkey
AND r2.H_Regionhashkey = x2.R_REgionhashkey
and x1.R_Name = 'AMERICA'
and x2.R_Name = 'AMERICA'
and (d_year = 1997 or d_year = 1998)
and (p_mfgr = 'MFGR#1'
or p_mfgr = 'MFGR#2')
group by d_year, t2.n_Name, p_category
order by d_year, t2.N_Name, p_category
```

Q4.3

```
SELECT d_year, s_city, p_brand, sum(U_Revenue -
lo_supplycost) as profit
from S_Umsatz,S_LineOrder,S_SUpplier, S_Date, S_Part, S_NATION t3,S_Nation t2, H_Supplier, L_LineOrder, H_Date, H_Part, Link_nationsupplier,H_Nation n1, H_Nation n2, link_nationregion ns1, link_nationregion ns2, H_region r1,H_Region r2, S_Region x1,S_Region x2, H_Customer, link_nationcustomer
where S_Umsatz.L_LOHashKey= L_LineOrder.L_LOHashKey
AND S_LineOrder.LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_DateHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND L_LineOrder.L_CustHashKey = H_Customer.H_CustHashKey
AND L_LineOrder.L_PartHashKey = H_Part.H_PartHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Part.H_PartHashKey = S_Part.PartHashKey
AND H_Supplier.H_SuppHashKey = S_Supplier.S_SuppHashKey
AND H_Supplier.H_SuppHashKey = link_nationsupplier.supplierhashkey
AND link_nationsupplier.nationhashkey = n1.n_nationhashkey
AND n1.N_NATIONHashKey = t2.N_NATIONHashKey
AND H_Customer.H_CustHashKey = link_nationcustomer.customerhashkey
AND link_nationcustomer.nationhashkey = n2.n_nationhashkey
AND n2.n_nationhashkey = ns2.nationhashkey
AND ns2.regionhashkey  = r2.H_Regionhashkey
```

```
AND r2.H_Regionhashkey = x2.R_REgionhashkey
and x2.R_NAme = 'AMERICA'
and t2.N_Name = 'UNITED STATES'
and (d_year = 1997 or d_year = 1998)
and p_category = 'MFGR#14'
group by d_year, s_city, p_brand
order by d_year, s_city, p_brand
```

C.3 Laufzeitmesswerte des erweiterten Data Vaults

Tabelle 9: Laufzeitmesswerte modifiziertes DV im Vergleich zum SSB und DV

Abfrage	Elapsed Time SSB	Elapsed Time DV	Elapsed Time modifiziertes DV
Q2.1	48813,863	58213,626	65498,265
Q2.2	30534,933	60260,707	70475,576
Q2.3	33560,03	47108,313	53025,868
Q3.1	79801,662	51817,587	84266,361
Q3.2	79411,981	48721,418	55674,205
Q4.1	80118,045	80213,251	140956,808
Q4.2	80811,57	133709,941	173961,07
Q4.3	51155,866	49349,53	58529,205

Anhang D: Data Vault mit Bridge-Tabelle

D.1 Abfragen des Data Vaults mit Bridge Tabelle

Q2.1

```
SELECT sum(U_Revenue), D_year, P_brand
FROM S_Umsatz, S_Date, S_Part, H_Supplier, L_LineOrder, H_Date, H_Part, bridge_renasu, H_region, S_Region
WHERE S_Umsatz.L_LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_DateHashKey
AND L_LineOrder.L_PartHashKey = H_Part.H_PartHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Part.H_PartHashKey = S_Part.PartHashKey
AND H_Supplier.H_SuppHashKey = bridge_renasu.Supplier_HashKey
AND bridge_renasu.Region_HashKey = H_Region.H_RegionHashKey
AND H_Region.H_Regionhashkey = S_Region.R_REgionhashkey
AND P_category = 'MFGR#12'
AND R_Name = 'AMERICA'
GROUP BY D_Year, P_Brand
ORDER BY D_year, P_brand;
```

Q2.2

```
SELECT sum(U_Revenue), D_year, P_brand
FROM S_Umsatz, S_Date, S_Part, H_Supplier, L_LineOrder, H_Date, H_Part, H_region, S_Region, bridge_renasu
WHERE S_Umsatz.L_LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_DateHashKey
AND L_LineOrder.L_PartHashKey = H_Part.H_PartHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Part.H_PartHashKey = S_Part.PartHashKey
AND H_Supplier.H_SuppHashKey = bridge_renasu.supplier_hashkey
AND bridge_renasu.region_hashkey = H_Region.H_Regionhashkey
```

```
AND H_Region.H_Regionhashkey = S_Region.R_REgionhashkey
AND P_Brand between 'MFGR#2221' and 'MFGR#2228'
AND R_Name = 'ASIA'
GROUP BY D_Year, P_Brand
ORDER BY D_year, P_brand;
```

Q2.3

```
SELECT sum(U_revenue), D_year, P_brand
FROM S_Umsatz, S_Date, S_Part, H_Supplier, L_LineOrder, H_Date, H_Part,bridge_renasu,
H_region, S_Region
WHERE S_Umsatz.L_LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_DateHashKey
AND L_LineOrder.L_PartHashKey = H_Part.H_PartHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Part.H_PartHashKey = S_Part.PartHashKey
AND H_Supplier.H_SuppHashKey = bridge_renasu.supplier_hashkey
AND bridge_renasu.region_hashkey = H_Region.H_Regionhashkey
AND H_Region.H_Regionhashkey = S_Region.R_REgionhashkey
AND P_Brand = 'MFGR#2339'
AND R_Name = 'EUROPE'
GROUP BY D_Year, P_Brand
ORDER BY D_year, P_brand;
```

Q3.1

```
SELECT distinct t2.N_Name, t3.N_Name, D_Year, sum(U_Revenue) as revenue
FROM S_Umsatz, S_Date, S_NATION t3,S_Nation t2, H_Supplier, L_LineOrder,
H_Date,H_Nation n1, H_Nation n2, link_nationregion , H_region r1,H_Region r2, S_Region
x1,S_Region x2, H_Customer, link_nationcustomer
where S_Umsatz.L_LOHashKey= L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_DateHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND L_LineOrder.L_CustHashKey = H_Customer.H_CustHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
```

```
AND H_Supplier.H_SuppHashKey = bridge_renasu.supplier_hashkey
AND bridge_renasu.nation_hashkey = n1.n_nationhashkey
AND bridge_renasu.region_hashkey = r1.H_RegionHashKey
AND n1.N_NATIONHashKey = t2.N_NATIONHashKey
AND n1.n_nationhashkey = ns1.nationhashkey
AND r1.H_Regionhashkey = x1.R_REgionhashkey
AND H_Customer.H_CustHashKey = link_nationcustomer.customerhashkey
AND link_nationcustomer.nationhashkey = n2.n_nationhashkey
AND n2.n_nationhashkey = link_nationregion.nationhashkey
AND link_nationregion.regionhashkey  = r2.H_Regionhashkey
AND r2.H_Regionhashkey = x2.R_REgionhashkey
AND n2.N_NATIONHashKey = t3.N_NATIONHashKey
and x1.R_Name = 'ASIA'
and x2.R_Name = 'ASIA'
and d_year >= 1992 and d_year <= 1997
group by t2.N_Name, t3.N_Name, D_Year
order by D_Year asc, Revenue desc;
```

Q3.2

```
SELECT C_City, S_City, D_Year, sum(U_Revenue) as revenue
from S_Umsatz, S_Date, S_Supplier, L_LineOrder,bridge_renasu, H_Date, H_Supplier,
S_Customer, H_Customer, link_nationcustomer, H_nation n1, H_Nation n2, S_Nation s1,
S_Nation s2
where S_Umsatz.L_LOHashKey= L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_DateHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND L_LineOrder.L_CustHashKey = H_Customer.H_CustHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Supplier.H_SuppHashKey = S_Supplier.S_SuppHashKey
AND H_Supplier.H_SuppHashKey = bridge_renasu.supplier_hashkey
AND bridge_renasu.nation_hashkey = n1.N_NATIONHashKey
AND n1.N_NATIONHashKey = s1.N_NATIONHashKey
AND H_Customer.H_CustHashKey = S_Customer.S_CustHashKey
AND H_Customer.H_CustHashKey = link_nationcustomer.customerhashkey
```

```
AND link_nationcustomer.nationhashkey = n2.N_NATIONHashKey
AND n2.N_NATIONHashKey = s2.N_NATIONHashKey
and s1.N_Name = 'UNITED STATES'
and s2.N_Name = 'UNITED STATES'
and d_year >= 1992 and d_year <= 1997
group by C_City, S_City, D_Year
order by D_Year asc, revenue desc;
```

Q4.2

```
SELECT d_year, t2.N_Name, p_category, sum(U_revenue -
lo_supplycost) as profit
from S_Umsatz,S_LineOrder, S_Date, bridge_renasu, S_Part, S_Nation t3, S_Nation t2, H_Supplier, L_LineOrder, H_Date, H_Part, Link_nationsupplier,H_Nation n1, H_Nation n2, link_nationregion ns1, link_nationregion ns2, H_region r1,H_Region r2, S_Region x1,S_Region x2, H_Customer, link_nationcustomer
where S_Umsatz.L_LOHashKey= L_LineOrder.L_LOHashKey
AND S_LineOrder.LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_DateHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND L_LineOrder.L_CustHashKey = H_Customer.H_CustHashKey
AND L_LineOrder.L_PartHashKey = H_Part.H_PartHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Part.H_PartHashKey = S_Part.PartHashKey
AND H_Supplier.H_SuppHashKey = bridge_renasu.supplier_hashkey
AND bridge_renasu.nation_hashkey = n1.n_nationhashkey
AND bridge_renasu.region_hashkey = r1.H_REgionhashkey
AND n1.N_NATIONHashKey = t2.N_NATIONHashKey
AND r1.H_Regionhashkey = x1.R_REgionhashkey
AND H_Customer.H_CustHashKey = link_nationcustomer.customerhashkey
AND link_nationcustomer.nationhashkey = n2.n_nationhashkey
AND n2.n_nationhashkey = ns2.nationhashkey
AND ns2.regionhashkey  = r2.H_Regionhashkey
AND r2.H_Regionhashkey = x2.R_REgionhashkey
and x1.R_Name = 'AMERICA'
```

```
and x2.R_Name = 'AMERICA'
and (d_year = 1997 or d_year = 1998)
and (p_mfgr = 'MFGR#1'
or p_mfgr = 'MFGR#2')
group by d_year, t2.n_Name, p_category
order by d_year, t2.N_Name, p_category
```

Q4.3

```
SELECT d_year, s_city, p_brand, sum(U_Revenue -
lo_supplycost) as profit
from S_Umsatz, S_LineOrder, S_SUpplier, S_Date, S_Part,bridge_renasu, S_Nation t2,
H_Supplier, L_LineOrder, H_Date, H_Part,H_Nation n1, H_Nation n2, link_nationregion ns1,
link_nationregion ns2,H_Region r2, S_Region x2, H_Customer, link_nationcustomer
where S_Umsatz.L_LOHashKey= L_LineOrder.L_LOHashKey
AND S_LineOrder.LOHashKey = L_LineOrder.L_LOHashKey
AND L_LineOrder.L_DateHashKey = H_DateHashKey
AND L_LineOrder.L_SuppHashKey = H_Supplier.H_SuppHashKey
AND L_LineOrder.L_CustHashKey = H_Customer.H_CustHashKey
AND L_LineOrder.L_PartHashKey = H_Part.H_PartHashKey
AND H_Date.H_DateHashKey = S_Date.D_DateHashKey
AND H_Part.H_PartHashKey = S_Part.PartHashKey
AND H_Supplier.H_SuppHashKey = S_Supplier.S_SuppHashKey
AND H_Supplier.H_SuppHashKey = bridge_renasu.supplier_hashkey
AND bridge_renasu.nation_hashkey = n1.n_nationhashkey
AND n1.N_NATIONHashKey = t2.N_NATIONHashKey
AND H_Customer.H_CustHashKey = link_nationcustomer.customerhashkey
AND link_nationcustomer.nationhashkey = n2.n_nationhashkey
AND n2.n_nationhashkey = ns2.nationhashkey
AND ns2.regionhashkey  = r2.H_Regionhashkey
AND r2.H_Regionhashkey = x2.R_REgionhashkey
and x2.R_NAme = 'AMERICA'
and t2.N_Name = 'UNITED STATES'
and (d_year = 1997 or d_year = 1998)
and p_category = 'MFGR#14'
```

```
group by d_year, s_city, p_brand
order by d_year, s_city, p_brand
```

D.2 Laufzeitmesswerte des Data Vaults mit Bridge Tabelle

Tabelle 10:Laufzeitmesswerte des Data Vaults mit Bridge-Tabelle im Vergleich zu den anderen DVs

Abfrage	Elapsed Time DV	Elapsed Time modifiziertes DV	Elapsed Time DV mit Bridge Table
Q2.1	58213,626	65498,265	61023,57
Q2.2	60260,707	70475,576	59175,107
Q2.3	47108,313	53025,868	45716,203
Q3.1	51817,587	84266,361	53707,113
Q3.2	48721,418	55674,205	54671,084
Q4.1	80213,251	140956,808	121316,011
Q4.2	133709,941	173961,07	134475,943
Q4.3	49349,53	58529,205	45227,081

Literatur

Abadi et al. 2009

Abadi, Daniel J.; Boncz, Peter A.; Harizopoulos, Stavros; *Column-oriented database systems*, in: Proceedings of the VLDB Endowment, 2, 2009, S. 1664–1665.

Chaudhuri et al. 1997

Chaudhuri, Surajit; Dayal, Umeshwar; *An overview of data warehousing and OLAP technology*, in: ACM Sigmod record, 26, 1997, S. 65–74.

Codd et al. 1993

Codd, Edgar F.; Codd, Sharon B.; Salley, Clynch T.; *Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate*, in: Codd and Date, 32, 1993.

Goeken 2007

Goeken, Matthias; *Entwicklung von data-warehouse-systemen: Anforderungsmanagement, modellierung, implementierung.* Springer-Verlag, 2007.

Inmon et al. 1994

Inmon, William H.; Hackathorn, Richard D.; *Using the data warehouse.* Wiley-QED Publishing, 1994.

Kimball et al. 2011

Kimball, Ralph; Ross, Margy; *The data warehouse toolkit: the complete guide to dimensional modeling.* John Wiley & Sons, 2011.

Klaßen et al. 2004

Klaßen, Henning; Andreas Buhmann, Betreut von; *DB-Caching,* 2004.

Köppen et al. 2014

Köppen, Veit; Saake, Gunter; Sattler, Kai-Uwe; *Data Warehouse Technologien.* mitp Verlags GmbH & Co. KG, 2014.

Levene et al. 2003

Levene, Mark; Loizou, George; *Why is the snowflake schema a good data warehouse design?*, in: Information Systems, 28, 2003, S. 225–240.

Linstedt et al. 2015

Linstedt, Dan; Olschimke, Michael; *Building a Scalable Data Warehouse with Data Vault 2.0: Implementation Guide for Microsoft SQL Server 2014.* Morgan Kaufmann, 2015.

Lusti 1998

Lusti, Markus; *Data Warehousing und Data Mining—Wege zur Eindämmung der Datenflut?*, in: , *Wertorientierte Unternehmensführung,* Springer, 1998, S. 285–309.

Margaret Rouse 2006

Margaret Rouse; *TechTarget's IT encyclopedia,* 2006, http://whatis.techtarget.com/definition/performance. Abgerufen am 05.04.2016.

O'Neil et al. 2009

O'Neil, Patrick; O'Neil, Elizabeth; Chen, Xuedong; Revilak, Stephen; *The star schema benchmark and augmented fact table indexing*, in: , *Performance evaluation and benchmarking,* Springer, 2009, S. 237–252.

Pendse 1995

Pendse, N.; *The FASMI definition for OLAP*, in: Business Intelli, 1995.

Peter et al. 1982

Peter, Tom; Waterman, Robert; *In search of excellence*, in: Lessons from Americans Best Running Companies. New York: Harper & Row, 1982.

Rick F. van der Lans 2015

Rick F. van der Lans; *Data Vault and Data Virtualization: Double Agility,* 2015. Abgerufen am 02.04.2016.

Srivastava et al. 2014

Srivastava, Kaushal; Srivastava, Sanjay; Sharma, Akhil; Pandey, Avinash; *Comparison of Star Schema and Snow Flake Schema using Telecommunication Database*, in: Star, 1, 2014.

TPC 2014

TPC; *TPC-H Documentation,* 2014.

Wrembel et al. 2007

Wrembel, Robert; Koncilia, Christian; *Data warehouses and OLAP: concepts, architectures, and solutions.* IGI Global, 2007.

Xu, Huan and Luo, Hao and He, Jieyue 2013

Xu, Huan and Luo, Hao and He, Jieyue (Hrsg.); *What-If Query Processing Policy for Big Data in OLAP System.* IEEE, 2013.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit selbständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Ort, Datum

Unterschrift