THM | CAMPUS FRIEDBERG | MND
TECHNISCHE HOCHSCHULE MITTELHESSEN
Mathematik, Naturwissenschaften und Datenverarbeitung

Masterarbeit

zum Thema

Performance-Nutzen-Analyse: Automatisierung eines Data Warehouse

Untersuchung am Beispiel des Datavault Builder auf verschiedenen Zielplattformen in der Azure Cloud

---

zur Erlangung des akademischen Grades Master of Science (M. Sc.)

Akademischer Betreuer: Professor Dr. Harald Ritz
Externer Betreuer: Arthur Arendt M. Sc.
OPITZ CONSULTING DEUTSCHLAND GmbH

vorgelegt von:	Kevin Ullrich Am Fahrenbach 9 61191 Rosbach v. d. H. Kevin.Ullrich@mnd.thm.de
Matrikelnummer:	5049655
Studiengang:	Wirtschaftsinformatik
Semesterzahl:	3
Semester der Abgabe:	Wintersemester 2021/2022
Abgabedatum:	20.10.2021

Sperrvermerk

Die vorgelegte Arbeit basiert auf internen, vertraulichen Daten und Informationen des Unternehmens OPITZ CONSULTING Deutschland GmbH. In diese Arbeit dürfen Dritte, mit Ausnahme der Gutachter und der befugten Mitglieder des Prüfungsausschusses, ohne ausdrückliche Zustimmung des Unternehmens und des Verfassers keine Einsicht nehmen. Eine Vervielfältigung und Veröffentlichung der Arbeit ohne ausdrückliche Genehmigung – auch auszugsweise – ist nicht erlaubt.

Abstract

In der vorliegenden Arbeit wird eine Performance-Nutzen-Analyse vorgenommen, untersucht am Beispiel des Datavault Builder zur automatischen Data Warehouse-Erstellung auf verschiedenen Zielplattformen (konkret: Azure SQL-Datenbank, Exasol und Snowflake) in der Azure Cloud. Die Aufgabenstellung im Zusammenhang mit der vorliegenden Arbeit besteht im Vergleich verschiedener Zielplattformen in Kombination mit dem Datavault Builder und darin, eine Aussage über die Vorteilhaftigkeit einer bestimmten Zielplattform für den Datavault Builder im Hinblick auf die Performance und Nutzung geben zu können. Zudem soll die Kenntnis der Arbeit es ermöglichen, die Thematik der Data Warehouse-Automatisierung tiefer zu durchdringen, um die jeweils bestmögliche Nutzung in Kundenprojekten zu garantieren. Dazu wird ein Benchmark in unterschiedlich großen Skalierungsausprägungen verwendet. Qualitative Aspekte werden mit Hilfe einer Nutzwertanalyse bewertet. Als Ergebnis wird deutlich, dass sich die Datenbank Exasol für alle drei TPC-H-Datensatzgrößen trotz des schlechteren Nutzwertes als vorteilhafter herausgestellt hat: zum einem deshalb, weil sie in jeder Skalierungsstufe die geringste Gesamtlaufzeit gegenüber der Azure SQL-Datenbank und Snowflake erzielt hat; zum anderen aus dem Grund, weil sie nur einen Bruchteil der Kosten der anderen beiden Zieldatenbanken verursacht. Die Azure SQL-Datenbank ist aufgrund ihrer langen Laufzeiten und den damit verbundenen Kosten weniger für den Einsatz mit dem Datavault Builder zu empfehlen. Snowflake hingegen ist zwar nicht viel langsamer als Exasol, verursacht aber erheblich höhere Kosten. Der Vorteil, den Snowflake jedoch gegenüber Exasol hat, ist darin zu sehen, dass Snowflake als Software-as-a-Service-Lösung bereitgestellt wird, bei der Snowflake alle Aspekte der Verwaltung abdeckt.

Gliederung

ABKÜRZUNGSVERZEICHNIS UND NACHWEIS EINER FORMEL VI

ABBILDUNGSVERZEICHNIS VIII

TABELLENVERZEICHNIS X

1 EINLEITUNG 1

1.1 EINFÜHRUNG IN DAS THEMA 1

1.2 ZIELSETZUNG UND FRAGESTELLUNG/ERKENNTNISINTERESSE 3

1.3 VORGEHENSWEISE 3

2 THEORETISCHE GRUNDLAGEN UND DEFINITIONEN 5

2.1 BEGRIFFLICHE EINORDNUNG 5

2.1.1 Data Warehouse 5

2.1.2 Data Warehousing 7

2.1.3 Data Warehouse-System 8

2.2 DATA WAREHOUSE-GRUNDKONZEPT 8

2.3 DATA WAREHOUSE-ARCHITEKTUREN 13

2.3.1 Data Warehouse Bus-Architektur 13

2.3.2 CIF-Architektur 15

2.3.3 Zentrales Data Warehouse 15

2.3.4 Föderierte Architektur 16

2.3.5 Data Vault 2.0-Architektur 17

2.4 DATA WAREHOUSE-ENTWICKLUNGSANSÄTZE 19

2.4.1 Wasserfallmodell 19

2.4.2 V-Modell 20

2.4.3 Agiles Modell 21

2.5 DATA WAREHOUSE-AUTOMATISIERUNG 24

2.5.1 Facetten der BI/DWH-Automatisierung 27

2.5.2 Data Warehouse-Automatisierungstools: Designansätze 29

2.5.3 Data Warehouse-Automatisierungstools: Motive für den Einstieg 29

3 VERWENDETE TECHNOLOGIEN UND WERKZEUGE 31

3.1 MICROSOFT AZURE 31

3.2 DOCKER 32

3.2.1 Images 33
3.2.2 Container 33
3.2.3 Volumes 33
3.3 DOCKER COMPOSE 34
3.4 DATAVAULT BUILDER 34
3.5 DATENBANKSERVER (ZIELPLATTFORMEN) 35
3.5.1 Azure SQL-Datenbank 35
3.5.2 Exasol 38
3.5.3 Snowflake 42
4 FALLSTUDIE 46
4.1 BESCHREIBUNG 46
4.2 VERWENDETE METHODEN 46
4.2.1 Benchmarking 48
4.2.2 Nutzwertanalyse 48
4.3 DER BENCHMARK 51
4.4 RAHMENBEDINGUNGEN 55
4.5 VERGLEICHSARCHITEKTUR 57
5 PERFORMANCE-NUTZEN-ANALYSE 59
5.1 PERFORMANCEANALYSE 59
5.1.1 Ingest Load-Performance 60
5.1.2 Raw Vault Load-Performance 63
5.1.3 Query-Performance (ohne Cache) 66
5.1.4 Query-Performance (mit Cache) 69
5.1.5 Gesamtkosten und -laufzeit 72
5.2 NUTZENANALYSE 74
6 GESAMTBEWERTUNG 79
7 ZUSAMMENFASSUNG UND FAZIT 83
LITERATURVERZEICHNIS XI
ANHANG 1 (Diagramme)
ANHÄNGE 2-5 (Nachweise etc.) ausschließlich in der THM-Hessenbox enthalten

Abkürzungsverzeichnis und Nachweis einer Formel

ADO ..... Active Data Objects
ANSI ..... American National Standards Institute
AVM ..... Azure Virtual Machine
AWS ..... Amazon Web Services
BI ..... Business Intelligence
CIF ..... Corporate Information Factory
CMMI ..... Capability Maturity Model Integration
CPU ..... Central Processing Unit
CSV ..... Comma Separated Values
DBA ..... Datenbankadministrator
DBMS ..... Data Base Management System
DTU ..... Database Transaction Unit
DVB ..... Datavault Builder
DVM ..... Data Vault Model
DWHA ..... Data Warehouse-Automatisierung
DWH ..... Data Warehouse
ER ..... Entity-Relation
ESD ..... Enterprise for Sensitive Data
ETL ..... Extrahieren, Transformieren und Laden
GB ..... Gigabyte
GCP ..... Google Cloud-Plattform
GF ..... Gewichtungsfaktor
GUI ..... Graphical User Interface
HDFS ..... Hadoop Distributed File System
I/O ..... Input/Output
JDBC ..... Java Database Connectivity
JSON ..... JavaScript Object Notation
LTS ..... Long Term Support
MPP ..... Massively Parallel Processing
ODBC ..... Open Database Connectivity
ODS ..... Operational Data Store
OLAP ..... Online Analytical Processing
OLTP ..... Online Transaction Processing

PaaS .....Platform-as-a-Service
PHI ..... Protected Health Information
RAM ..... Random Access Memory
SaaS ..... Software-as-a-Service
SCD .....Slowly Changing Dimension
SDLC ..... Software Development Life Cycle
SPOT ..... Single Point of Truth
SQL .....Structured Query Language
SSD ..... Solid State Drive
TB ..... Terabyte
TPC .....The Transaction Processing Performance Council
vCPU .....Virtual Centralized Processing Unit
VM ..... Virtuelle Maschine
VPS ..... Virtuelle Private Snowflake
XML .....Extensible Markup Language
YAML .....Yet Another Markup Language

Formel *: Additive Funktion 51

Abbildungsverzeichnis

Abbildung 1: Übersicht über die Struktur der Arbeit 4
Abbildung 2: Multidimensionales Star-Schema des Kaufauftrags 10
Abbildung 3: Snowflake-Schema des Kaufauftrags 10
Abbildung 4: Data Vault-Modell mit Hubs, Links und Satellites für ein Luftfahrtszenario 12
Abbildung 5: Kimballs Data Warehouse-Architektur 14
Abbildung 6: Bill Inmons Data Warehouse-Architektur 15
Abbildung 7: Zentralisierte Data Warehouse-Architektur 16
Abbildung 8: Föderierte Data Warehouse-Architektur 17
Abbildung 9: Referenzarchitektur Data Vault 2.0 18
Abbildung 10: Wasserfallmodell 20
Abbildung 11: Das V-Modell 21
Abbildung 12: Lebenszyklus agiles Modell 23
Abbildung 13: Iteration bei einem agilen Ansatz 23
Abbildung 14: Vorteile der Data-Warehouse-Automatisierung 24
Abbildung 15: Einsparungspotenzial von DWHA 25
Abbildung 16: Umfang der Automatisierung 26
Abbildung 17: Überblick über Exasol und seine Funktionen 39
Abbildung 18: Exasol-Kernarchitektur 41
Abbildung 19: Snowflake-Kernarchitektur 43
Abbildung 20: Geschäftsumfeld des TPC-H-Benchmark 54
Abbildung 21: Das TPC-H-Schema 54
Abbildung 22: Vergleichsarchitektur 58
Abbildung 23: TPC-H Datavault Modell 59
Abbildung 24: Ingest Load-Performance 1 GB 60
Abbildung 25: Ingest Load-Performance 5 GB 61
Abbildung 26: Ingest Load-Performance 10 GB 62
Abbildung 27: Raw Vault Load-Performance 1 GB 63
Abbildung 28: Raw Vault Load-Performance 5 GB 64
Abbildung 29: Raw Vault Load-Performance 10 GB 65
Abbildung 30: Query-Performance (ohne Cache) 1 GB 66
Abbildung 31: Query-Performance (ohne Cache) 5 GB 67
Abbildung 32: Query-Performance (ohne Cache) 10 GB 68

Abbildung 33: Query-Performance (mit Cache) 1 GB _____69
Abbildung 34: Query-Performance (mit Cache) 5 GB _____70
Abbildung 35: Query-Performance (mit Cache) 10 GB _____71
Abbildung 36: Gesamtkosten und -laufzeit 1 GB _____72
Abbildung 37: Gesamtkosten und -laufzeit 5 GB _____73
Abbildung 38: Gesamtkosten und -laufzeit 10 GB _____74
Abbildung 39: Ergebnis der Nutzwertanalyse _____79

Tabellenverzeichnis

Tabelle 1: Skalierungsstufen der Zieldatenbanken 56
Tabelle 2: Bewertungskriterien 75
Tabelle 3: Paarweiser Vergleich der Kriterien 76
Tabelle 4: Ermittlung der Gewichtungsfaktoren 76
Tabelle 5: Bewertungsskala 77
Tabelle 6: Ergebnis der Nutzwertanalyse 78

1 Einleitung

1.1 Einführung in das Thema

Als digitale Transformation wird die kontinuierliche Weiterentwicklung digitaler Technologien bezeichnet, welche die Gesellschaft und die Wirtschaft nachhaltig prägt. Die digitale Transformation eröffnet Unternehmen eine neue Welt der Möglichkeiten, um mehr und bessere Geschäfte zu tätigen. Obwohl die IT-Strategie in Bezug auf die digitale Transformation von Unternehmen zu Unternehmen variiert, ist eine wesentliche Gegebenheit jeder IT-Strategie die Nutzung und die Analyse von Daten, die generiert werden. Daten ermöglichen es Unternehmen, tiefere Erkenntnisse über ihre Kunden, Produkte, Konkurrenten und neue Märkte zu erlangen, wodurch sich ein Wettbewerbsvorteil ergibt. Um jedoch die Vorteile dieser neuen Welt der Möglichkeiten zu nutzen, ist es unabdingbar, die immensen Datenmengen effektiv und effizient zu verwalten.

Seit mehr als 25 Jahren spielen Data Warehouse-Systeme eine entscheidende Rolle beim Sammeln von Daten und ihrer Transformation in wertvolle Informationen. Ein Data Warehouse-System bezeichnet eine Sammlung von Systemkomponenten und Einzeldatenbanken, die in einem mehrstufigen Prozess aus einer Vielzahl von Quellsystemen abgeleitet werden (vgl. Lehner, 2003: 9 f.). Auf diese Weise können die Anwender Entscheidungen auf der Grundlage fundierter Geschäftsfakten ableiten und müssen diese nicht vornehmlich auf der Basis von Intuitionen treffen.

Es ist weithin anerkannt, dass ein Data Warehouse (kurz DWH) eine kritische Komponente für ein datengesteuertes Unternehmen ist (vgl. Eckerson, 2015: 2) und Teil der Informationssystemstrategie des Unternehmens wird, die einen signifikanten Einfluss auf das Geschäft hat (vgl. Inmon, et al., 2008: 11). Die ganzheitliche Sicht auf die Daten, die von DWHs geliefert werden, ermöglicht es Organisationen, ihr Geschäft derart optimiert zu kontrollieren, dass sie ihre Effizienz und Wettbewerbsfähigkeit erhöhen können. Dies ist entscheidend für Prozesse wie z. B. das Kundenbeziehungs- und Betrugsmanagement (vgl. Inmon, et al., 2008: 11). DWHs stellen eine zentrale Komponente zur Unterstützung des Entscheidungsprozesses einer Organisation dar und sind die Quelle von Entscheidungsunterstützungssystemen. Zudem bieten sie eine Möglichkeit, den Endnutzern die für die Entscheidungsfindung erforderlichen Informationen zur Verfügung zu stellen (vgl. Schnider, et al., 2016: 1).

Der Aufbau eines DWH ist ein mühsames Unterfangen. DWHs sind zeitaufwendig, teuer und im produktiven Betrieb nur schwer zu ändern (vgl. Wells, 2014). Es gibt viele Aktivitäten rund um den Aufbau eines DWH, die den Prozess arbeitsintensiv und zeitaufwendig machen

(vgl. Rahman, et al., 2015: 1), wie z. B. Anforderungserfassung und -analyse, Quelldatenanalyse, Quell-Ziel-Mapping, Datentransformationslogik, Extrahieren, Transformieren, Laden (ETL), Prozesse, Datenanalyse, Design und Bereitstellung. Zudem ist die Integration verschiedener Informationen erforderlich, die sich üblicherweise auf unterschiedlichen Systemen, in unterschiedlichen Formaten und Volumina innerhalb der Organisation befinden (vgl. Eckerson, 2015: 2).

Die meisten DWHs folgen einem Wasserfall-Systementwicklungszyklus (engl. Software Development Life Cycle, kurz SDLC) (vgl. Rainardi, 2008: 43), der relativ zeitaufwendig und zu unflexibel ist, um sich schnell an geschäftliche Anforderungen und Veränderungen anzupassen (vgl. Evelson und Angel, 2016: 2). Aufgrund dieser Situation werden zu einem bestimmten Zeitpunkt im Lebenszyklus eines Projekts Ressourcenprobleme auftauchen, weshalb das Projektteam gezwungen sein wird, Überstunden zu erbringen, um den Projektzeitplan einzuhalten (vgl. Rahman, et al., 2015: 1). Im Allgemeinen wird auf eine solche Situation mit dem Hinzufügen von Ressourcen zum Projekt reagiert, was aber letztendlich keinen produktiven Weg darstellt, um das Problem zu lösen, wie Brooks in seinem Buch anmerkt: „Adding manpower to a late software project makes it later" (Brooks, 1975: 25). Während das Projektteam Zeit und Mühe investiert, um neuen Teammitgliedern den Projektkontext zu erklären, sinkt die Produktivität, was zur Folge hat, dass Projekte hinter dem Zeitplan zurückbleiben (vgl. Rahman, et al., 2015: 1 f.).

Viele der in Unternehmen eingesetzten DWH-Systeme sind zudem in die Jahre gekommen. Ihre Anfänge liegen zum Teil über zehn Jahre zurück; die damaligen Konzepte sind jedoch nach wie vor nur wenig verändert im Einsatz. Wie jeder andere Unternehmensbereich, so unterliegen auch DWH-Lösungen einem immer stärkeren Zeit-, Kosten- und Qualitätsdruck. Die geforderte Effizienz stellt eine große Herausforderung dar, insbesondere für Unternehmen mit bestehenden, historisch gewachsenen DWH-Systemen, Organisationen und Prozessen. Herkömmliche DWHs werden folglich mit den wachsenden Anforderungen der Fachabteilungen und der Geschäftsleitung nicht mehr Schritt halten können. Die Lösung dieses Problems liegt in der Automatisierung von Data Warehousing mit agilen Entwicklungsmethoden, hochgradiger Standardisierung und Automatisierung. Die Werkzeuge zur Data Warehouse-Automatisierung (kurz DWHA) setzen an diesen Punkten an (vgl. Blaschek, 2017: 1; Stauffer, 2017: 1; sowie Zeiler, 2017: 1 f.). Trotz des stetig wachsenden Marktes ist das Wissen über die vorhandenen Möglichkeiten noch recht rudimentär. Die mit der vorliegenden Arbeit verbundene Intention besteht nicht zuletzt darin, mehr Verständnis für die Chancen der DWHA zu schaffen.

1.2 Zielsetzung und Fragestellung / Erkenntnisinteresse

Unter DWHA versteht man den Prozess der Beschleunigung und der Reduktion manueller Vorgänge der DWH-Entwicklungszyklen bei gleichzeitiger Gewährleistung von Qualität und Konsistenz. Die Opitz Consulting GmbH verwendet zur Automatisierung von DWH den Datavault Builder (kurz DVB) der 2150 GmbH. Die Aufgabenstellung im Zusammenhang mit der vorliegende Arbeit besteht darin, verschiedene Zielplattformen in Kombination mit dem DVB zur automatischen DWH-Erstellung zu vergleichen und eine Aussage über die Vorteilhaftigkeit einer Zielplattform für den DVB im Hinblick auf die Performance und Nutzung geben zu können. Zudem soll die Kenntnis der Arbeit es ermöglichen, die Thematik der DWHA tiefer zu durchdringen, um die jeweils bestmögliche Nutzung in Kundenprojekten zu garantieren.

Als Methode wird in der vorliegenden Arbeit eine Performance-Nutzen-Analyse vorgenommen – untersucht am Beispiel des DVB zur automatischen DWH-Erstellung auf verschiedenen Zielplattformen (konkret: Azure SQL-Datenbank, Exasol und Snowflake) in der Azure Cloud. Die mit der Arbeit verbundene Intention besteht darin, die Ergebnisse der Performance-Nutzen-Analyse in klar strukturierter Weise darzustellen und komplexe Sachverhalte nachvollziehbar zu erläutern. Zum Zweck einer umfassenden Betrachtung werden sowohl quantitative als auch qualitative Faktoren betrachtet. Mit den erzielten Ergebnissen soll auf folgende Fragen eine Antwort gefunden werden:

- In welcher Kombination ist der DVB am performantesten?
- In welcher Kombination ist der DVB am einfachsten zu handhaben?
- In welcher Kombination ist der DVB preislich am günstigsten?
- Welchen Nutzen bieten die unterschiedlichen Zielplattformen?
- Bestehen weitere Anbindungsmöglichkeiten, beispielsweise zu Azure Analysis Services und/oder Power BI?

1.3 Vorgehensweise

Die Arbeit gliedert sich in sieben Kapitel. Nach einer Klärung notwendiger informationstechnischer Grundlagen, die essenziell für das Verständnis der Arbeit sind (Kapitel 2), werden die verwendeten Technologien und Werkzeuge, die zur Gegenüberstellung der Performance- und Nutzenanalyse herangezogen werden, beschrieben (Kapitel 3). Daran anschließend wird die der Arbeit zugrunde liegende Fallstudie vorgestellt: Diese Fallstudie wird zunächst beschrieben (Kapitel 4, Abschnitt 1), anschließend wird auf die verwendeten Methoden eingegangen

(Kapitel 4, Abschnitt 2), der Benchmark wird erläutert (Kapitel 4, Abschnitt 3), es werden die Rahmenbedingungen der Performance-Nutzen-Analyse festgelegt (Kapitel 4, Abschnitt 4) und die Vergleichsarchitektur vorgestellt (Kapitel 4, Abschnitt 5). Da einerseits eine ausschließliche Performancebetrachtung als zu einseitig anzusehen ist und andererseits eine rein qualitative Vorgehensweise dem Rationalitätsprinzip der Wirtschaftlichkeit widerspricht, wird in Kapitel 5 eine Performance-Nutzen-Analyse durchgeführt. Im Anschluss daran erfolgt die Beantwortung der in Abschnitt 1.2 genannten Fragen durch die Bewertung der Performance-Nutzen-Analyse (Kapitel 6). Kapitel 7 stellt eine Zusammenfassung der Arbeitsergebnisse und ein Fazit dar. Die Struktur der Arbeit ist grafisch in Abbildung 1 dargestellt:

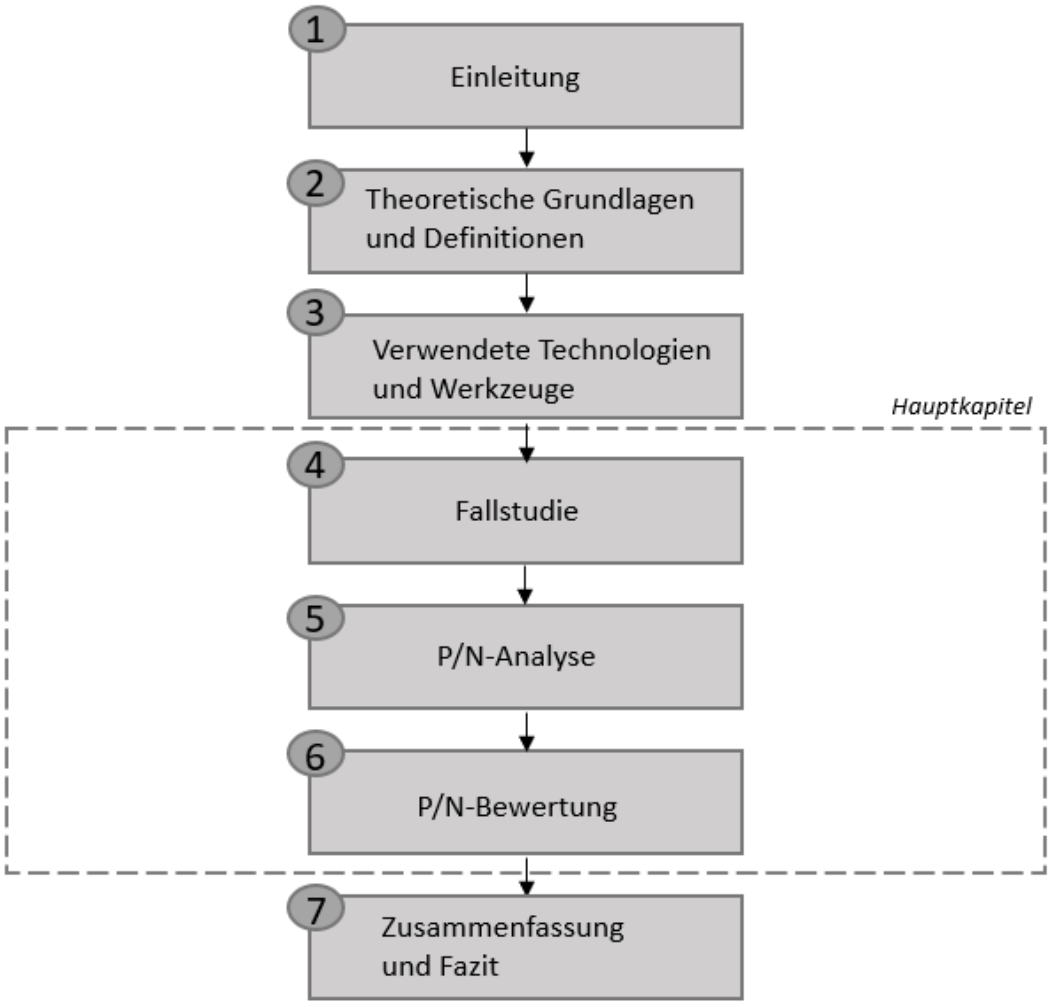


Abbildung 1: Übersicht über die Struktur der Arbeit

(Quelle: Eigene Darstellung)

2 Theoretische Grundlagen und Definitionen

In diesem Kapitel werden die theoretischen Grundlagen zum Thema ‚Data Warehousing' erläutert. Hierzu werden die erforderlichen Begriffsdefinitionen und Erläuterungen zu Data Warehouse (DWH), Data Warehousing, DWH-System, zu DWH-Architekturen, zu DWH-Entwicklungsansätzen sowie zur DWHA gegeben. Dabei wird sich auf die wesentlichen Grundlagen beschränkt, da jedes Thema für sich allein Fachbücher füllt und eine größere Ausführlichkeit den Rahmen dieser Arbeit sprengen würde.

2.1 Begriffliche Einordnung

Die Begriffe DWH, Data Warehousing und DWH-System tauchen in vielen Fachzeitschriften, in Fachwörterbüchern und anderen Fachpublikationen mit zum Teil unterschiedlichen Definitionen und Interpretationen auf. Um in dieser Arbeit eine einheitliche Definition mit einheitlichen Fachbegriffen darzustellen, werden die in der Literatur augenscheinlich am häufigsten verwendeten und zitierten Definitionen verwendet.

2.1.1 Data Warehouse

In den 1990er-Jahren führte William Harvey Inmon erstmals das Konzept eines DWH ein, weshalb er auch als „Vater des Data Warehousing" bezeichnet wird. Ein DWH entsteht mit dem Ziel, den Entscheidungsprozess einer Organisation zu unterstützen und steht im Zentrum der Entscheidungsunterstützungssysteme. Es stellt eine der einfachsten und intuitivsten Methoden dar, um die ihm gespeicherten Informationen den Benutzern zur Verfügung zu stellen, die sie zum Treffen von Entscheidungen benötigen. W. H. Inmon hat bereits 1996 ein DWH wie folgt definiert und charakterisiert:

> „A data warehouse is a subject-oriented, integrated, nonvolatile,
> and time-variant collection of data in support of management's decisions."
> (Inmon, 2005: 29)

Ein DWH hat seiner Meinung nach vier Eigenschaften, die die Entscheidungsfindung unterstützen. Die Eigenschaften Fachorientierung (engl. subject-oriented), integrierte Datenbasis (engl. integrated), nicht flüchtige Datenbasis (engl. nonvolatile) und historische Daten (engl. time-variant) werden im Folgenden kurz skizziert:

- Fachorientierung

  Der Zweck der Datenbasis fungiert nicht zur Durchführung einzelner Transaktionen, Aufträge oder Buchungen, sondern dient der Auswertung ganzer Themenbereiche wie ‚Produkte' und ‚Kunden' durch Kennzahlen wie Kosten und Umsatz. Eine zweckneutrale Darstellung für bestimmte Anwendungen erfordert die logische und physische Organisation der Daten.

- Integrierte Datenbasis

  Die Datenverarbeitung erfolgt auf integrierten Daten aus vielen verschiedenen Quellen und externen Systemen, die in dieselben Formate konvertiert werden und denselben Regeln und Konventionen folgen. Dabei bedeutet ‚Integration' die Funktionalität und Fähigkeit, Daten aus verschiedenen Systemen miteinander in Verbindung zu bringen. Daten werden aus verschiedenen inkompatiblen Systemen mit unterschiedlichen Datenmodellen extrahiert, bereinigt und dann zu einem integrierten Datenbestand verknüpft.

- Nicht flüchtige Datenbasis

  Die Datenbasis ist als stabil zu betrachten. In einer DWH-Umgebung stellen die vorhandenen Daten eine Art ‚Schnappschuss' eines bestimmten Zeitpunktes oder Raums dar. Sie werden periodisch aktualisiert (stündlich, täglich, wöchentlich oder monatlich, je nach Anwendung). Einmal in das DWH-System eingebrachte Daten werden nicht mehr entfernt oder verändert.

- Historische Daten

  Um Vergleiche über die Zeit zu ermöglichen, werden die Daten so angelegt, dass eine Historisierung der extrahierten Zustände und Daten über die Zeit erreicht wird. Die Historisierung kann aus systemtechnischen Gründen, z. B. Speicherkapazität, begrenzt sein.

Das DWH kann als ein zentrales Datenlager verstanden werden, das aufbereitete Daten aus inkompatiblen Datenbanken einschließlich Datenmodellen aus einer Datenbank oder aus mehreren autonomen Datenbanken enthält. Für Unternehmen stellt es oft den hochwertigen Analysepunkt (Single Point of Truth kurz SPOT) dar, der alle relevanten Informationen sammelt und archiviert. Kommen neue Informationen hinzu, werden die alten Informationen beibehalten, was zu einer Historisierung der Daten führt. Um die riesigen Datenmengen zu halten, ist es wichtig, sinnvolle Granularitäts- und Verdichtungsebenen zu planen und zu implementieren. Der große Vorteil eines DWH ist darin zu sehen, dass die Daten aus verschiedenen Datenquellen bereinigt, integriert und analysiert werden können, ohne die Funktion der

operativen Systeme, die als Quellen dienen, zu beeinträchtigen. Die physische Trennung zwischen DWH und Quelldaten ermöglicht insbesondere die Modellierung der Daten im Hinblick auf analytische Anwendungen (vgl. Farkisch, 2011: 5-10; Köppen, et al., 2014: 1 f.; sowie Bauer und Günzel, 2013: 5-11).

2.1.2 Data Warehousing

Data Warehousing, auch DWH-Prozess genannt, beschreibt einen dynamischen Prozess, beginnend mit dem Datenerfassungsprozess, über die Speicherung bis hin zur Analyse und Auswertung der Daten. Dieser beschriebene Prozess ist notwendig, um ein DWH-System zu planen, aufzubauen und zu betreiben. Data Warehousing umfasst die folgenden Aktivitäten:

1. die Extraktion der relevanten Daten aus den Quellsystemen,
2. die Transformation und Bereinigung der Daten aus den Quellsystemen,
3. das Laden der Daten in das DWH,
4. die dauerhafte Speicherung der Daten im DWH,
5. die Bereitstellung der integrierten Daten für Analysezwecke,
6. die Analyse und Auswertung der Daten.

Die ersten drei Punkte werden auch zusammenfassend als ETL-Prozess bezeichnet, der im Folgenden kurz erläutert wird (vgl. Farkisch, 2011: 8; und Bauer und Günzel, 2013: 9).

Die Abkürzung ETL steht für Extract, Transform und Load, was im Deutschen Extrahieren, Transformieren und Laden bedeutet. ETL beschreibt einen aus mehreren Einzelschritten bestehenden Prozess, mit dem Daten aus mehreren Datenquellen in eine Datenbank oder ein DWH integriert werden können. Bei der Extraktion werden durch regelmäßige Selektion relevante Änderungsdaten aus den Quellen der Datenbank oder des DWH zur Verfügung gestellt. Die Transformationsphase schließt sich an die Extraktionsphase an und passt die extrahierten Daten an die vorgegebene Zielstruktur an, indem Aspekte der Formatierung, der Bereinigung fehlerhafter Daten, der Entfernung von Duplikaten sowie der Gruppierung, Sortierung und Aggregation von Daten festgelegt werden. Nachdem die extrahierten Daten bereinigt und transformiert wurden, werden sie in die Datenbank oder das DWH geladen. Ziel des ETL-Prozesses ist es, die integrierten Daten für die weitere Verarbeitung vorzubereiten und verfügbar zu machen (vgl. Farkisch, 2011: 59-69; und Luber und Litzel, 2018).

2.1.3 Data Warehouse-System

In Anlehnung an Wolfgang Lehner wird, wie bereits einführend dargelegt, ein DWH-System definiert als eine Sammlung von Systemkomponenten und Einzeldatenbanken, die in einem mehrstufigen Prozess aus einer Vielzahl von Quellsystemen abgeleitet werden (vgl. Lehner, 2003: 9 f.). Der Begriff DWH-System bezieht sich in diesem Zusammenhang auf die gesamte technische Infrastruktur zur Gewinnung, Speicherung und Auswertung von Daten. Der Unterschied zwischen einem DWH-System und anderen Informationssystemen besteht darin, dass bei einem DWH-System Daten, die einmal in das DWH geladen wurden, nicht mehr verändert werden. Neue Daten können in das DWH geladen werden, ohne dass die bestehenden Daten überschrieben werden (vgl. Farkisch, 2011: 7; und Bauer und Günzel, 2013: 8).

2.2 Data Warehouse-Grundkonzept

In diesem Abschnitt wird das grundlegende Konzept des DWH vorgestellt, welches zum Verständnis der DWHA benötigt wird. Es wird darauf hingewiesen, dass eine vollständige und detaillierte Einführung in das Thema „DWH“ den Rahmen dieser Arbeit sprengen würde. Zudem wird davon ausgegangen, dass der interessierte und fachkundige Leser mit den Grundlagen eines DWH vertraut ist, weshalb sich auf das Wesentliche in der thematischen Darlegung beschränkt wird.

Für eine ausführliche Zusammenfassung der Prinzipien und Ansätze sei auf die Bücher von William Inmon und Ralph Kimball verwiesen. Eine grundlegende Einführung in Datavault kann sich zudem mittels der Lektüre des Buchs von Daniel Linstedt und Michael Olschimke angeeignet werden (Linstedt und Olschimke, 2016).

Ein DWH-System besteht aus verschiedenen Komponenten, wovon jeder eine spezifische Funktion in der Architektur zugewiesen ist. Die Quellsysteme sind die Datenquellen für das DWH-System und können intern oder extern vorliegen. Bei den internen Quellen kann es sich um ein operatives System handeln, dessen Funktion die Erfassung der täglichen Geschäftstransaktionen ist (vgl. Kimball und Ross, 2013: 18 f.). Externe Quellen sind Informationsquellen, die von außerhalb der Organisation kommen und strukturierte oder unstrukturierte Informationen enthalten können.

Die für das DWH relevanten Daten werden aus den verschiedenen, zum Teil heterogenen Datenquellen, durch einen Extraktionsprozess gewonnen und in einen separaten Arbeitsbereich (Staging Area) geladen. In diesem Arbeitsbereich werden die Daten für die Integration nach zuvor definierten Regeln transformiert. Die transformierten und zu integrierenden Daten

werden in eine Basisdatenbank geladen (vgl. Farkisch, 2011: 56 f.). Dieser Vorgang von Extraktion über Transformation bis hin zum Laden wird als ETL-Prozess bezeichnet (vgl. Abschnitt 2.1.2).

Das DWH ist das Zentrum einer DWH-Architektur und damit das Haupt-Repository. Ungeachtet der unterschiedlichen Ansätze von William Inmon und Ralph Kimball zum Thema ‚Data Warehousing' stimmen beide darin überein, dass das DWH ein fachorientiertes, nichtflüchtiges, integriertes, historisiertes Repository ist, das den Entscheidungsprozess einer Organisation unterstützt (vgl. Inmon, 2005: 29). Es gibt jedoch noch weitere Repositories, wie z. B. den Operational Data Store (ODS), der Teil der DWH-Architektur sein kann. Mehr Informationen zum ODS sind bei (Kimball, et al., 2011: 124) zu finden.

Ein weiteres Repository in der Architektur ist der Data Mart. Data Marts sind spezielle, analyseorientierte, aus der Datenbasis abgeleitete Datenbanken mit einer Ausrichtung auf das jeweilige Anwendungsthema (vgl. Farkisch, 2011: 59). Über eine Reihe von Anwendungen und Tools greifen Benutzer des DWH-Systems auf Informationen zu, die im DWH und in den Data Marts gespeichert sind. Dieser Vorgang wird als Datenzugriff (engl. Data Access) bezeichnet.

Ebenso sind die Art der Modellierung und deren Konzepte eine wichtige Entscheidung beim Aufbau einer DWH-Architektur. Die gängigsten Modellierungsarten sind die dimensionale Modellierung und der Data Vault. Die dimensionale Modellierung ist eine logische Entwurfstechnik, die darauf abzielt, die Informationen für den Benutzer intuitiv zu strukturieren und eine hohe Performance zu ermöglichen. Dieser Modelltyp verweist auf eine zentrale Tabelle mit einem mehrteiligen Schlüssel, die als Faktentabelle bezeichnet wird, sowie eine Reihe von Tabellen, die als Dimensionen bezeichnet werden. Die dimensionale Modellierung hat zwei Hauptansätze:

1. Star-Schema: Das Star-Schema ist ein dimensionales Modellierungsdesign, bei dem die Faktentabelle das Zentrum bildet, und die Dimensionstabellen, die die Geschäftsobjekte repräsentieren, sternförmig mit der Faktentabelle verknüpft sind. Mit Abbildung 2 wird zum besseren Verständnis das Star-Schema visualisiert:

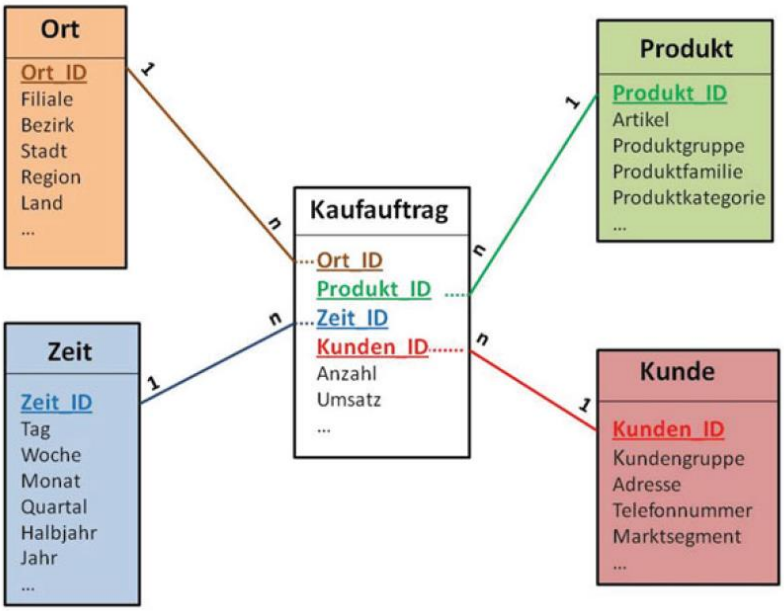


Abbildung 2: Multidimensionales Star-Schema des Kaufauftrags

(Quelle: Farkisch, 2011: 29)

2. Snowflake-Schema: Das Snowflake-Schema ist dem Star-Schema sehr ähnlich, da beide die Dimensionen um eine Faktentabelle herum organisieren. Der Hauptunterschied besteht darin, dass in einem Snowflake-Schema die Dimensionen normalisiert sind. Abbildung 3 dient ebenfalls dem besseren Verständnis dieses Konzepts:

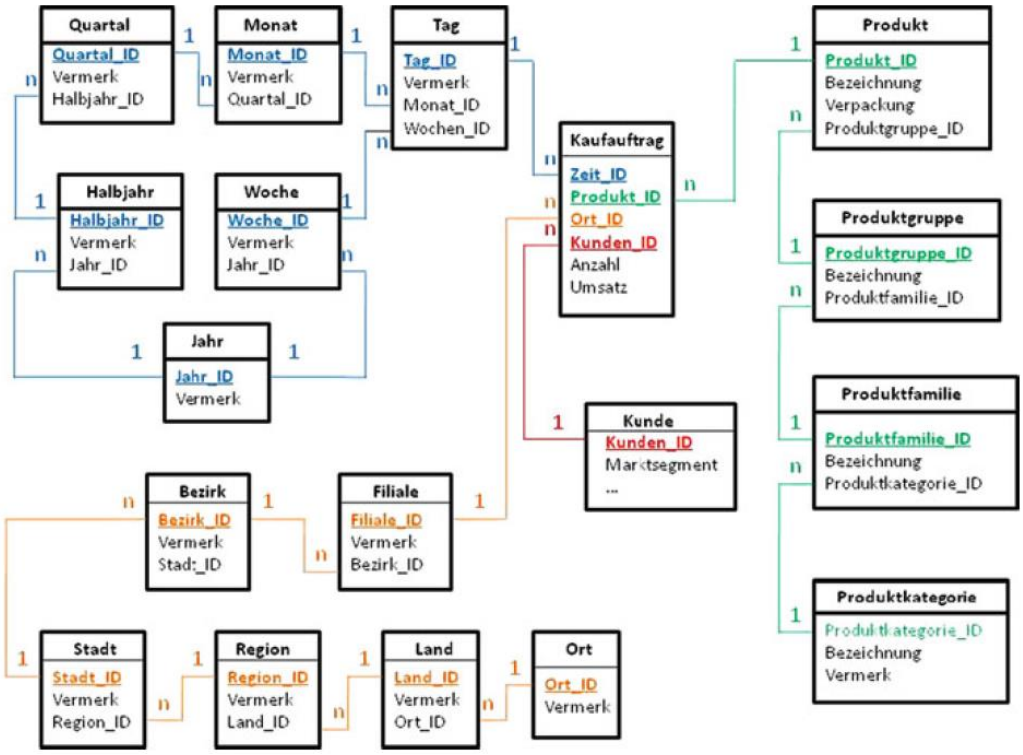


Abbildung 3: Snowflake-Schema des Kaufauftrags

(Quelle: Farkisch, 2011: 30)

Ein DWH bietet in der Regel die Option, die Entwicklung aller möglichen Arten von Daten im Zeitverlauf zu beobachten. Dazu müssen die Daten nicht nur für den aktuellen Zeitpunkt zur Verfügung stehen, wie es bei operativen Systemen der Fall ist, sondern für den gesamten Betrachtungszeitraum. Slowly Changing Dimension (SCD) (dt. eine „sich langsam ändernde Dimension“) ist die am weitesten verbreitete Technik, um Dimensionsänderungen über die Zeit zu erfassen. Eine Slowly Changing Dimension ist eine Dimension, die sowohl aktuelle als auch historische Daten in Zeitverläufen in einem DWH speichert und verwaltet. Näheres zur SCD findet sich in (Franke, 2008; und Duvuri, 2020).

Eine andere Art der Modellierungstechnik, die für den Entwurf von DWHs verwendet wird, ist Data Vault. „A DVM (Data Vault Model) is a detail-oriented, historical tracking, and uniquely linked set of normalized tables that support one or more functional areas of business“ (Inmon, et al., 2019: 141). Bei dieser Technik handelt es sich um einen hybriden Ansatz zwischen der 3NF (Dritte Normalform) und dem Sternschema. Datavault ist flexibel, skalierbar, konsistent und an die Anforderungen heutiger Unternehmen anpassbar. Die Nutzung von Data Vault (2.0) dient dazu, die Prozesse zur Erstellung von BI-(Business Intelligence)Systemen für Unternehmen auf wiederholbare, konsistente und skalierbare Weise zu verbessern und gleichzeitig eine nahtlose Integration mit neuen Technologien (z. B. NoSQL-Umgebungen) zu ermöglichen (vgl. Inmon, et al., 2019: 139 und 141).

Ein Data Vault-Modell basiert auf drei Hauptkonzepten: Hubs, Links und Satellites. Hubs sind Entitäten, die Geschäftsschlüssel speichern. Links sind Entitäten, die zwei oder mehr Hubs miteinander verbinden. Satelliten sind Entitäten, die Informationen über die Geschäftsschlüssel und damit über die Hubs speichern. Zudem speichern Satelliten aber auch Informationen zu den Links (vgl. Linstedt und Olschimke, 2016: 91). Abbildung 4 zeigt das Beispiel eines Data-Vault-Modells für ein Luftfahrtszenario. Es besteht ein Link zwischen der Fluggesellschaft, der Flugnummer und den Flughafen-Hubs, die den Flug repräsentieren. Die fünf Satelliten speichern Informationen über die Fluggesellschaft, den Flugstatus, den Flug und die Flughäfen.

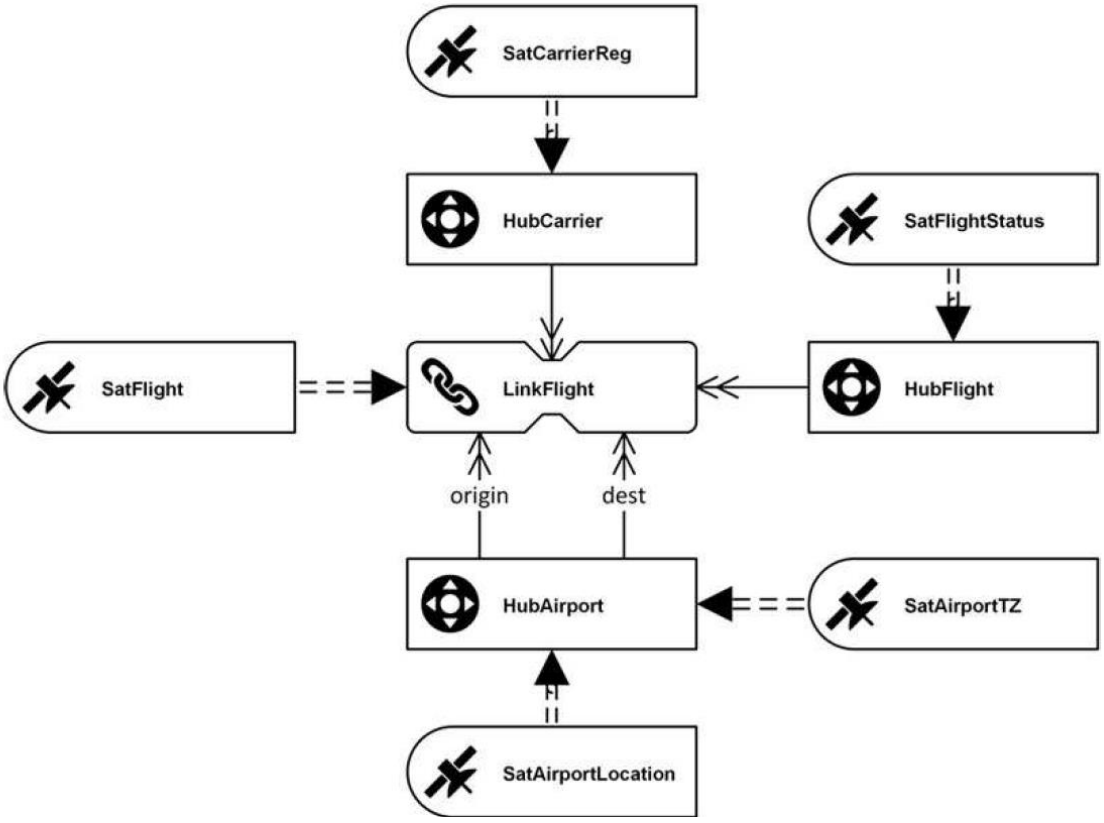


Abbildung 4: Data Vault-Modell mit Hubs, Links und Satellites für ein Luftfahrtszenario

(Quelle: Linstedt und Olschimke, 2016: 93)

Der Schwerpunkt von Data Vault als Modellierungstechnik zum Entwerfen logischer und physischer DWHs ist als Data Vault 1.0 bekannt. Das Konzept entwickelte sich über die Zeit zu Data Vault 2.0 weiter und umfasst fünf Komponenten, die als entscheidend für den Erfolg von BI und Data Warehousing gelten: Modellierung (Design für Leistung und Skalierbarkeit); Methodik (Agil und Scrum); Architektur (Big-Data und NoSQL); Implementierung (musterbasiert, Automatisierung und Generierung von CMMI[1] Level 5) (vgl. Linstedt und Olschimke, 2016: 11).

Da die Automatisierung als entscheidend für den Erfolg der DWH-Implementierung angesehen wird, enthalten die meisten DWHA-Tools heutzutage Data Vault-Best-Practices, die sofort einsatzbereit sind. Einige der DWHA-Tools verwenden Data Vault 1.0 und einige Data Vault 2.0.

[1] CMMI (Capability Maturity Model Integration) ist ein Ansatz für eine effektivere und effizientere Arbeit von Entwicklungsteams. CMMI basiert auf dem Capability Maturity Model, das sich aus einer Checkliste von kritischen Erfolgsfaktoren entwickelt hat, und bietet eine Reihe von Best Practices, wie bei der Verwaltung, Bewertung und Überwachung von Softwareentwicklungsprozessen am besten vorgegangen wird (vgl. Software Engineering Institute, 2011: 15).

2.3 Data Warehouse-Architekturen

Bei der Menge an Informationen, die jede Minute generiert werden, besteht die Herausforderung für Organisationen nicht nur darin, Informationen für den Entscheidungsprozess zu sammeln, sondern sie so effektiv zu verwalten, dass sie einen Wettbewerbsvorteil bieten. Doch wie entwickelt man ein am besten dafür geeignetes DWH-System?

Es gibt viele Möglichkeiten, ein DWH-System zu entwickeln, doch alle basieren auf einer Data Warehousing-Architektur, die definiert werden kann als eine Reihe von Designrichtlinien, Beschreibungen und Vorschriften für die Integration verschiedener Module, wie z. B. DWH, Data Mart, Operational Data Store, Staging Area, ETL-Tools sowie Analyse- und Reportingtools (vgl. Lans, 2012: 47).

Verursacht durch die Herausforderungen der Verwaltung großer Datenmengen verschiedener Datentypen – strukturiert, semistrukturiert, unstrukturiert und unterschiedlicher Größen – entsteht der Bedarf an neuen Data Warehousing-Architekturen, die in der Lage sind, diesen Anspruch zu genügen. In den nun folgenden Abschnitten werden die am häufigsten vorkommenden Data Warehousing-Architekturen vorgestellt.

Als die bisher populärsten Architekturen gelten: Ralph Kimballs Architektur, bekannt als DWH Bus-Architektur (Zwei-Schichten-Architektur), die Corporate Information Factory (CIF), auch als Hub-and-Spoke-Architektur bekannt, von Bill Inmon, Claudia Imhoff und Ryan Sousa (Drei-Schichten-Architektur), das zentralisierte DWH und die föderierte Architektur. In jüngerer Zeit ist eine weitere Architektur entstanden: die Data Vault-Architektur.

2.3.1 Data Warehouse Bus-Architektur

Ralph Kimball hat eine häufig verwendete Zwei-Schichten-Architektur vorgestellt (vgl. Kimball, et al., 2011: 114). In dieser Architektur, die in Abbildung 5 dargestellt ist, gibt es nur zwei Schichten, die Teil des DWH-Systems sind.

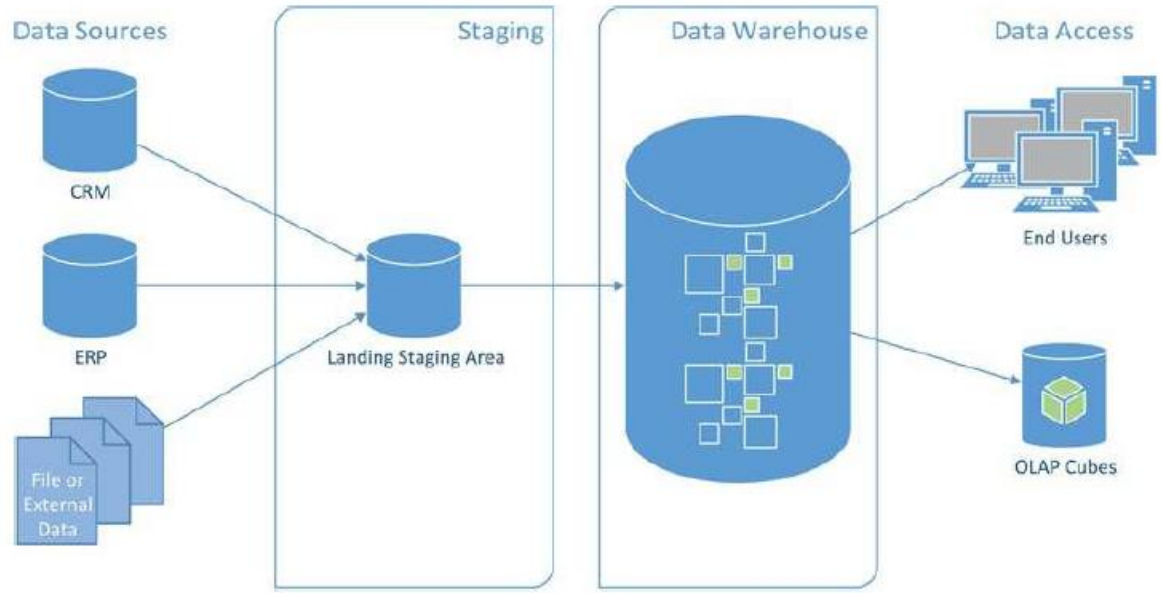


Abbildung 5: Kimballs Data Warehouse-Architektur

(Quelle: Linstedt und Olschimke, 2016: 13)

Die Rohdaten aus den Quellsystemen werden in den Stage-Bereich geladen, um eine exakte Kopie aller Daten zu erhalten, die in das DWH geladen werden sollen. Der Hauptnutzen des Stage-Bereichs besteht darin, die Anzahl der Operationen auf dem Quellsystem und die Zeit zum Extrahieren der Daten zu reduzieren. Die Tabellen im Stage-Bereich werden entsprechend den Tabellen im Quellsystem modelliert. Ein Stage-Bereich wird benötigt, wenn die Transformationen komplex sind und nicht „on-the-fly“ durchgeführt werden können oder wenn Daten von mehreren Quellsystemen zu unterschiedlichen Zeiten ankommen (vgl. Rainardi, 2008: 33). Nachdem die Daten in den Stage-Bereich geladen wurden, sollen, so Kimballs Vorschlag, die Daten in das DWH geladen werden. Das DWH ist dem Dimensionsmodell nachempfunden und besteht aus Data Marts, welche die Geschäftsprozesse repräsentieren. Das Modell wurde 1996 erstmals von Kimball vorgestellt. Es ist ein De-facto-Standard, der von Geschäftsanwendern mittels Analysetools, wie OLAP-Frontends oder -Engines, leicht abgefragt werden kann. Da es sich um eine logische Verknüpfung von konformen Data Marts handelt, müssen Geschäftsregeln vor der DWH-Schicht implementiert werden, um die Datensätze zu konformieren und abzugleichen. Datenzugriffsanwendungen verwenden das dimensionale Modell, um dem Benutzer die gewünschten Informationen zu präsentieren und Ad-hoc-Analysen zu ermöglichen. Ein Vorteil einer zweischichtigen Architektur besteht darin, dass es im Vergleich zu anderen Architekturen relativ unkompliziert ist, einen dimensionalen Speicher aus den Quelldaten zu erstellen. Ein Nachteil ist jedoch, dass der Aufbau eines zweiten dimensionalen Modells aus denselben Quelldaten komplexer ist, da

die Daten erneut aus dem Staging geladen werden müssen. (vgl. Linstedt und Olschimke, 2016: 12 f.)

2.3.2 CIF-Architektur

William Inmon befürwortet eine Drei-Schichten-Architektur für das DWH, die Corporate Information Factory (CIF). Zum besseren Verständnis der Architektur wird sie in Abbildung 6 vorgestellt, wobei die wichtigsten Begriffe erläutert werden:

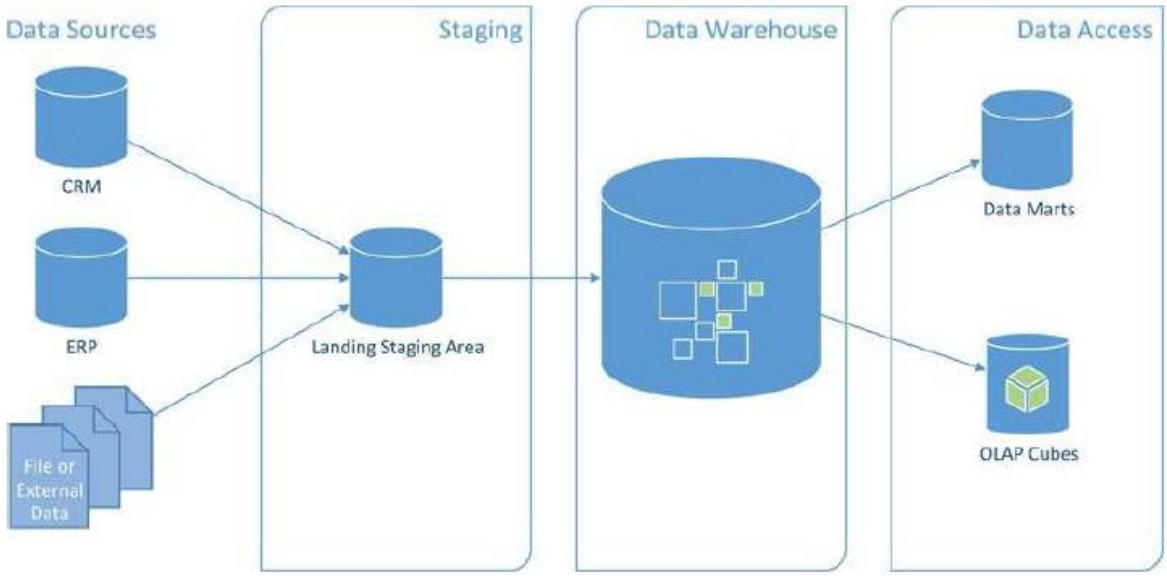


Abbildung 6: William Inmons Data Warehouse-Architektur

(Quelle: Linstedt und Olschimke, 2016: 14)

Das Enterprise DWH bildet das Kernstück der CIF-Architektur und ist ein integriertes Repository von atomaren Daten (vgl. Song, 2009: 688). Die Struktur der Informationen im DWH basieren auf ER(Entity-Relation)-Modellierungstechniken, liegen in der 3NF vor und werden in einer Realtime-Datenbank gespeichert. Die Data Marts sind abteilungsbezogene und zusammengefasste Ansichten von Informationen, die auf atomarer Ebene im DWH gespeichert, mit dimensionalen Techniken modelliert und der bevorzugte Zugangspunkt für analytische Anwendungen sind.

2.3.3 Zentrales Data Warehouse

Sollen nur Daten aus einer Anwendungsdomäne für die analytischen Anwendungen berücksichtigt werden, kann der Verzicht auf Data Marts eine Alternative sein. Stattdessen kann ein zentrales DWH aufgebaut werden, auf dem die Analysen direkt durchgeführt werden. Neben dem Analyseaspekt hat ein Core DWH auch eine Sammel-, Integrations- und Verteilungsfunktion, wodurch die Qualitätssicherung gewährleistet ist. Durch die Architektur eines

zentralen DWH sind Betrieb und Wartung des Systems zunächst einfacher zu realisieren als bei einer Siloarchitektur. Aufgrund der wenigen Komponenten sind Änderungen in den Datenstrukturen auch direkt für alle Anwendungen sichtbar. Dies bedeutet, dass vorgenommene Änderungen im Rahmen eines Change-Prozesses schnell verfügbar sind. Komplexere Lösungen stoßen jedoch aus Performance- und Administrationsgründen schnell an ihre Grenzen (vgl. Hahne, 2014: 8 f.). Der architektonische Ansatz ist in Abbildung 7 visualisiert:

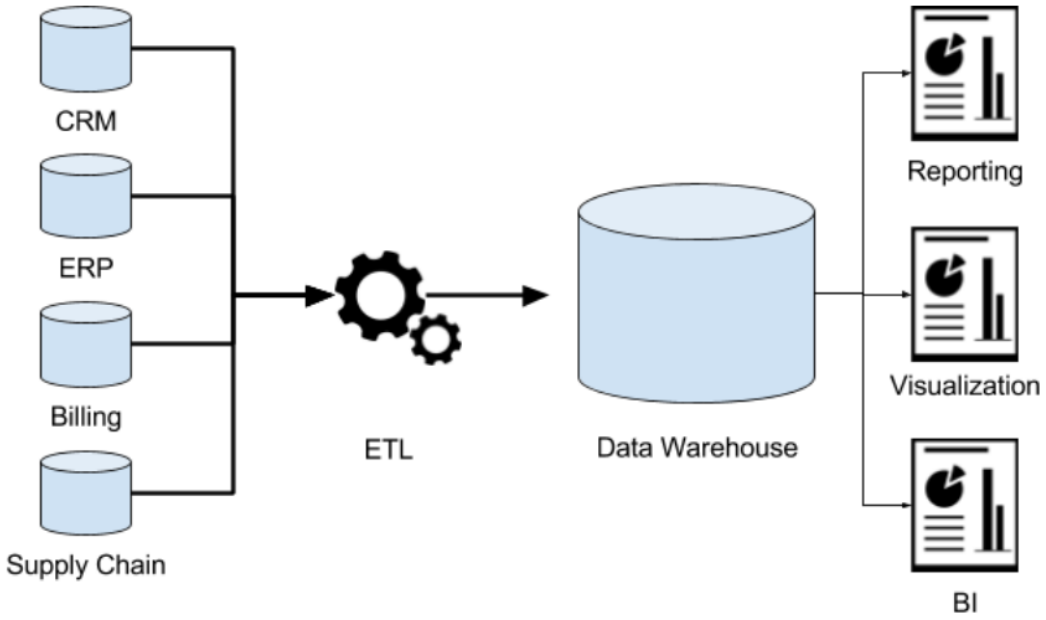


Abbildung 7: Zentralisierte Data Warehouse-Architektur

(Quelle: Dremio, 2021)

2.3.4 Föderierte Architektur

In der föderierten Architektur sind alle Berichts- und Analyseanwendungen mit einer eigenen Schicht verbunden, die mehrere Datenquellen aus heterogenen und autonomen Datenspeichern integriert, die unter anderem DWHs, Produktionsdatenbanken, Data Marts und persönliche Datenspeicher sein können (vgl. Lans, 2012: 47).

Das Fundament der föderierten Architektur ist das gemeinsame Geschäftsmodell der Organisation. Dabei ist das Geschäftsmodell ein sich ständig weiterentwickelndes semantisches Verständnis des Unternehmens in übereinstimmendem Verständnis aller Geschäftsbereiche. Dieses gemeinsame Geschäftsmodell ist der Ausgangspunkt für einen iterativen Prozess zum Aufbau der unterschiedlichen geschäftsintelligenten Subsysteme, basierend auf einem gemeinsamen Stage-Bereich (vgl. Rajan und Abhishek, 2019: 3). Eine föderierte DWH-Architektur ist in Abbildung 8 dargestellt:

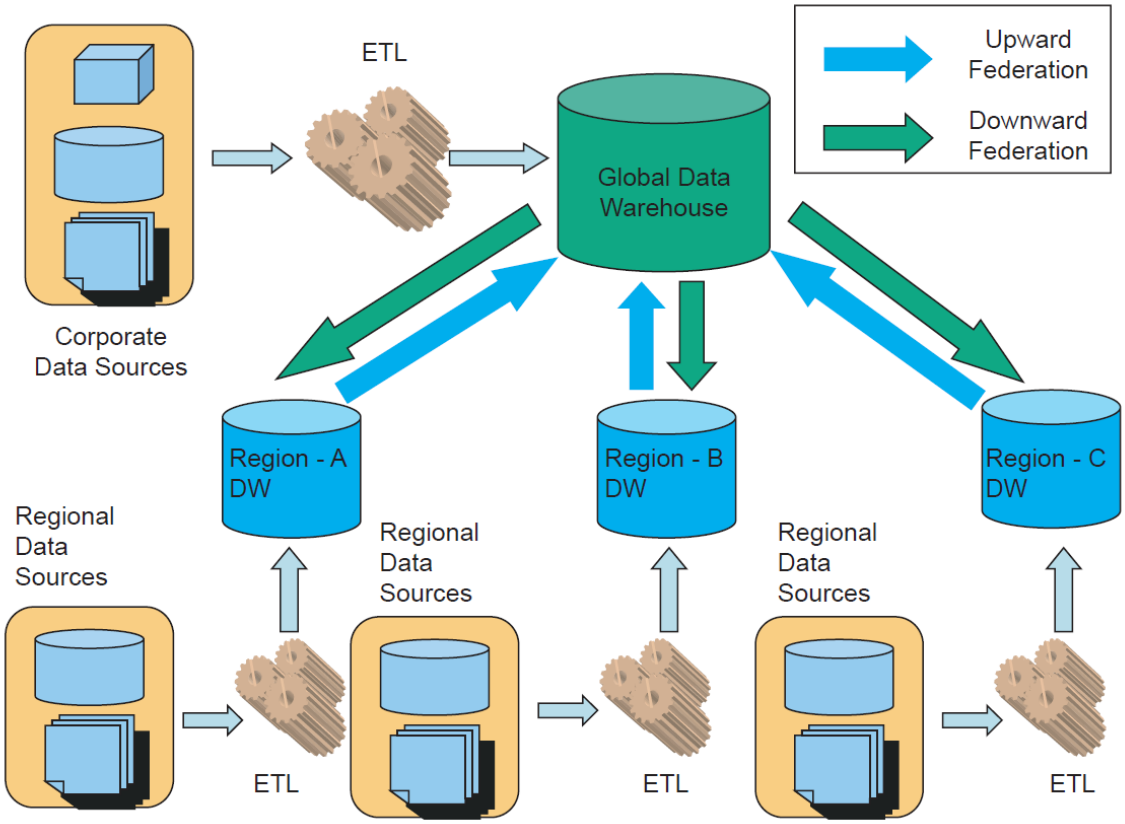


Abbildung 8: Föderierte Data Warehouse-Architektur

(Quelle: Rajan und Abhishek, 2019: 5)

In der obigen Abbildung verfügt eine Organisation über regionale DWHs für regionale analytische Anforderungen und über ein globales DWH für die Unternehmensanforderungen. Der Unterschied zwischen den beiden Systemen (regional und global) basiert auf der Art der Daten, die in jedem System enthalten sind. Das globale DWH speichert größtenteils verdichtete Daten für Unternehmensanalysen. Regionale DWHs speichern Daten für regionale analytische Anforderungen, die im Allgemeinen detailliertere Informationen enthalten. Die Referenzdaten bilden die Integrationsplattform für die DWHs. Der Datenfluss von regionalen DWHs zum globalen DWH wird als Upward-Federation bezeichnet, der Datenfluss vom globalen DWH zu den regionalen DHWs als Downward-Federation. Für die Datenkonsistenz und -integrität sollten die Referenzdaten in den verschiedenen DWHs einheitlich sein. Eine einheitliche und konsistente Definition der Referenzdaten in den beteiligten DWHs gewährleistet den SPOT in der gesamten föderierten Architektur.

2.3.5 Data Vault 2.0-Architektur

Die Architektur von Data Vault 2.0 basiert auf drei Schichten und umfasst zum einen die Stage-Schicht, welche die Rohdaten aus den Quellsystemen abgreift, weiterhin die als Data

Vault 2.0-Modell entworfene DWH-Schicht (siehe Abschnitt 2.2), und drittens die Informationsbereitstellungsschicht, mit Information Marts als Sternschema und anderen Strukturen (vgl. Linstedt und Olschimke, 2016: 22).

Der Hauptunterschied dieser Architektur zu anderen DWH-Architekturen besteht darin, dass die Geschäftsregeln, die harte oder weiche Geschäftsregeln sein können, je nachdem, ob es sich um technische oder geschäftliche Transformationen handelt, in den Information Marts umgesetzt werden. Die Architektur unterstützt Batch- oder Echtzeit-Laden sowie unstrukturierte NoSQL-Datenbanken (vgl. Linstedt und Olschimke, 2016: 31). Der Business Vault ist optional; er dient zum Speichern von Informationen, auf welche Geschäftsregeln angewendet wurden, und ist Teil der DWH-Schicht. Abbildung 9 stellt die Data Vault 2.0-Architektur zum besseren Verständnis graphisch dar:

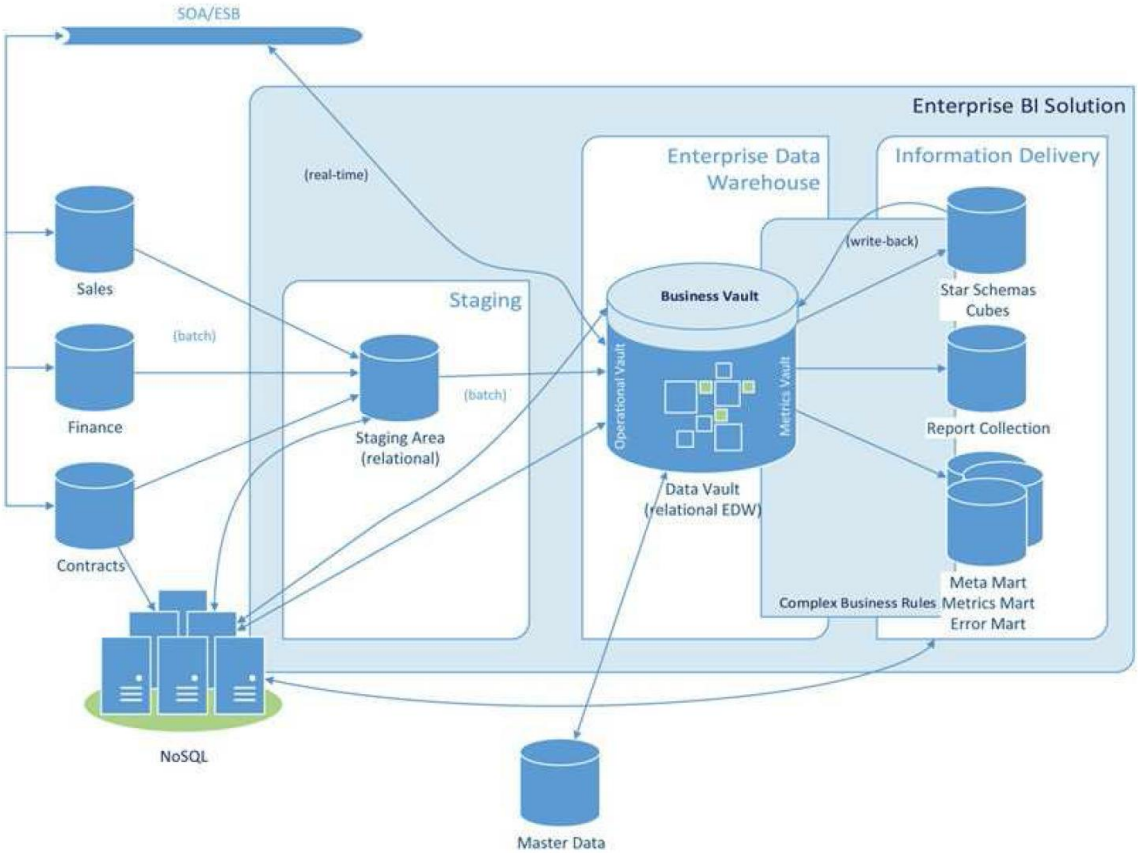


Abbildung 9: Referenzarchitektur Data Vault 2.0

(Quelle: Linstedt und Olschimke, 2016: 28)

2.4 Data Warehouse-Entwicklungsansätze

Voraussetzung für die Entwicklung eines funktionierenden DWH ist der Einsatz eines geeigneten Vorgehensmodells, das eine strukturierte, nachvollziehbare und variierbare Steuerung des gesamten Entwicklungsprozesses gewährleistet. Daher ist es notwendig, sich vor Beginn eines solchen Projekts für ein geeignetes Vorgehensmodell zu entscheiden.

Gerade im Bereich der Softwareentwicklung sind die zu beherrschenden Prozesse, d. h. die zusammenhängende Abfolge einzelner Aktivitäten, oftmals sehr komplex. Um diese Komplexität beherrschen zu können, haben sich in den letzten Jahren und Jahrzehnten verschiedene Vorgehensmodelle etabliert. Sie dienen im Wesentlichen dazu, die logische und zeitliche Abfolge der jeweils auszuführenden Tätigkeit, die Zuordnung von Personen und Rollen zu diesen Tätigkeiten sowie die Definition der zu erzielenden Ergebnisse festzulegen (vgl. Goeken, 2006: 52-54).

Die Wahl des jeweils richtigen Vorgehensmodells für die Entwicklung von Softwarelösungen ist keine einfache Aufgabe (vgl. Balaji und Murugaiyan, 2012: 26); deshalb ist es wichtig, die bestehenden Vorgehensmodelle sowie deren Vor- und Nachteile zu kennen, um fundiert entscheiden zu können, welches Vorgehen am besten zum Ziel des Projekts führt. In den folgenden Unterabschnitten werden die Hauptunterschiede zwischen den drei Vorgehensmodellen: Wasserfallmodell, V-Modell und Agiles Modell hervorgehoben.

2.4.1 Wasserfallmodell

Das Wasserfallmodell ist ein sequenzielles Vorgehensmodell, bei dem die Ausgaben jeder Phase die Eingaben für die nächste Phase liefern (vgl. Balaji und Murugaiyan, 2012: 27). Zum besseren Verständnis stellt Abbildung 10 ein Wasserfallmodell dar:

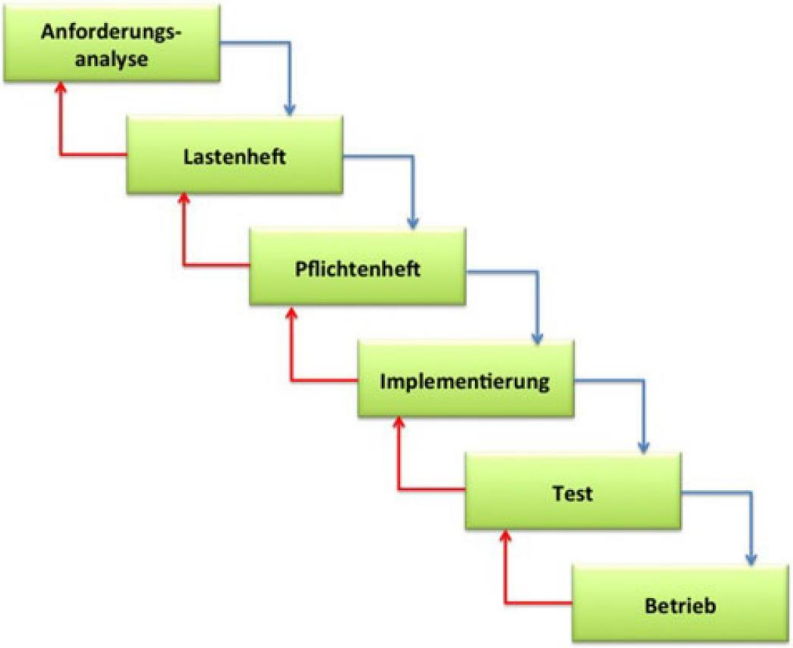


Abbildung 10: Wasserfallmodell

(Quelle: Vivenzio und Vivenzio, 2013: 6)

Die einzelnen Schritte laufen in einer festen Reihenfolge wie die Kaskaden eines Wasserfalls ab und sind in der Regel nicht umkehrbar. Als Erweiterung ist hier jedoch die Möglichkeit vorgesehen, zurückzuspringen (in diesem Fall fließt das Wasser unnatürlich bergauf). Wird ein Fehler erkannt, kann man einen Schritt zurückgehen und nach den Ursachen suchen. Entweder wird der Fehler behoben oder es muss ein weiterer Schritt zurück gemacht werden, was sehr zeitaufwendig und arbeitsintensiv sein kann! Das Wasserfallmodell bedient sich eines iterativen Vorgehens.

Es handelt sich um ein stark top-down-orientiertes und dokumentengetriebenes Vorgehensmodell, welches auf der Grundannahme von unveränderlichen Anforderungen basiert, die Projektphasen sind sehr starr. Das Modell besticht durch seine Einfachheit – vorausgesetzt, dass die Anforderungen präzise formuliert sind. Es ist leicht zu erlernen und auf jeden Fall besser als eine Software beziehungsweise ein DWH ohne Konzept zu planen. Wegen seiner Einfachheit eignet es sich eher für kleine und überschaubare Projekte. Bei größeren Projekten sollte man jedoch auf andere Modelle zurückgreifen. Der Aufwand bei sich ändernden Rahmenbedingungen kann im Wasserfallmodell sehr groß werden und nimmt dann dementsprechend viel Zeit in Anspruch (vgl. Vivenzio und Vivenzio, 2013: 5 f.)

2.4.2 V-Modell

Das V-Modell ist eine Erweiterung des Wasserfallmodells und zeigt die Beziehung zwischen jeder Entwicklungsphase und der zugehörigen Testphase. Im Gegensatz zum

Wasserfallmodell ist das V-Modell nicht linear aufgebaut; die Stufen verlaufen nach Abschluss der Implementierungsphase wieder nach oben (siehe Abbildung 11) (vgl. Balaji und Murugaiyan, 2012: 28).

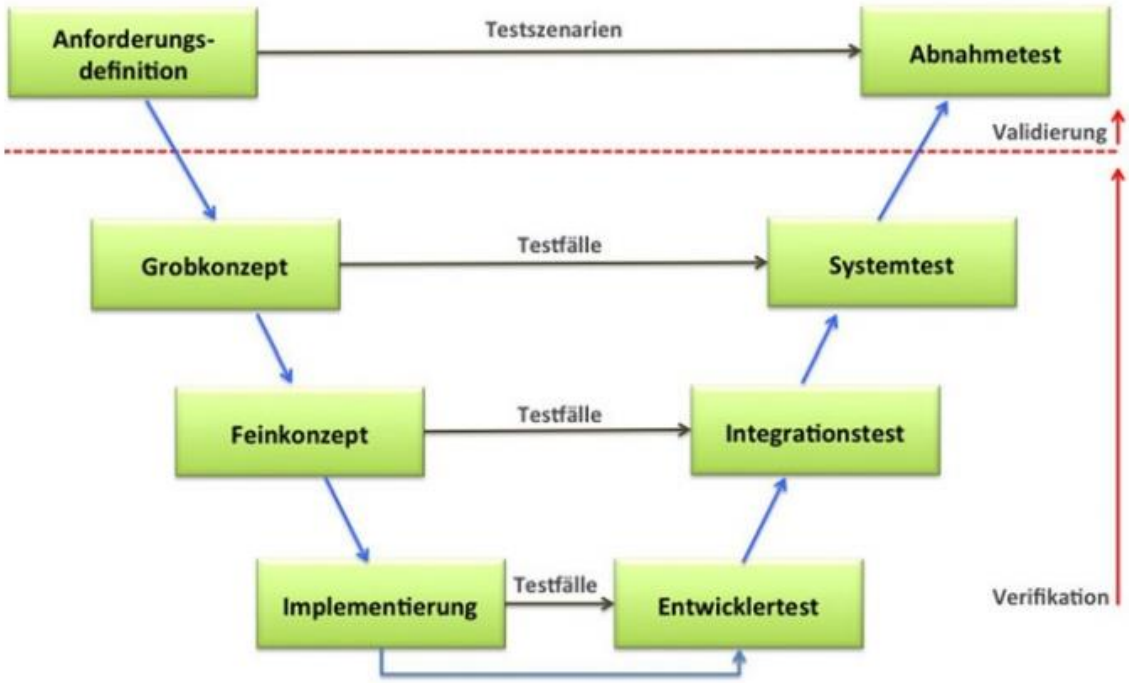


Abbildung 11: Das V-Modell

(Quelle: Vivenzio und Vivenzio, 2013: 7)

Die einzelnen Phasenergebnisse sind verbindliche Vorgaben für die nächst tiefere Projektphase. Der nach unten führende linke Zweig für die Spezifikationsphasen schließt mit der Implementierungsphase ab. Eine Erweiterung des Wasserfallmodells sind die nachfolgenden Testphasen, die im rechten, nach oben führenden Zweig dargestellt werden. Die Spezifikationsphasen werden den Testphasen gegenübergestellt, wodurch sich in der Darstellung ein charakteristisches ‚V' ergibt, das dem Modell auch seinen Namen gab. Diese Gegenüberstellung soll zu einer möglichst hohen Testabdeckung führen, da die Spezifikationen der jeweiligen Entwicklungsphasen die Grundlage für die Tests (Testfälle) in den entsprechenden Teststufen sind. Für das V-Modell im Allgemeinen sind die Anzahl der Phasen und auch deren Bezeichnungen unterschiedlich dargestellt, jedoch immer mit 1:1-Vergleich von Entwurfs- und Testphasen (vgl. Vivenzio und Vivenzio, 2013: 7)

2.4.3 Agiles Modell

Agile Methoden akzeptieren grundsätzlich, dass Planung, Verlauf und Durchführung eines Projekts schwer vorhersehbar sind, und versuchen, darauf flexibel zu reagieren. Der Fokus

liegt auf dem wirtschaftlichen Ziel des Projekts; genaue Anforderungen werden in der Regel lediglich skizziert und später durch das mit der Umsetzung beauftragte Team weiter spezifiziert. Gibt es spezifische Vorgaben, wie z. B. zu verwendende Technologien oder bereits vorhandene Systemkomponenten, so werden diese vom Auftraggeber als Bedingung vorgegeben und sind vom umsetzenden Team zwingend einzuhalten. Merkmale des agilen Vorgehens sind vor allem:

- iteratives Vorgehen auf Basis eines klar definierten Rahmens;
- vollständige Fertigstellung von Arbeitspaketen oder Produktinkrementen inklusive Integration, Test und Freigabe;
- Lieferung produktiver, nutzbarer Teilergebnisse nach einer oder mehreren Iterationen;
- Umsetzung von Teilprojekten oder Arbeitspaketen nach einer festgelegten Priorität;
- Wiederholung von Iterationen bis zur Fertigstellung des Gesamtprojekts;
- Analyse, Planung und Umsetzung durch das Team statt durch eine einzelne Person, wie z. B. einen Projektleiter.

Durch das iterative Vorgehen (siehe Abbildungen 12 und 13) bieten agile Methoden die Möglichkeit, während der gesamten Projektphase schnell auf kurzfristige Änderungen zu reagieren. Diese Änderungen können sich auf den Projektverlauf, die Anforderungen, die äußeren Bedingungen und auch den Endtermin beziehen. Mittels Auswertungen nach jeder Iterationsphase optimiert das Team seine Prozesse, Methoden und Arbeitsweisen, und berücksichtigt die identifizierten Verbesserungen bereits in der nächsten Iterationsphase (vgl. Bauer und Günzel, 2013: 408).

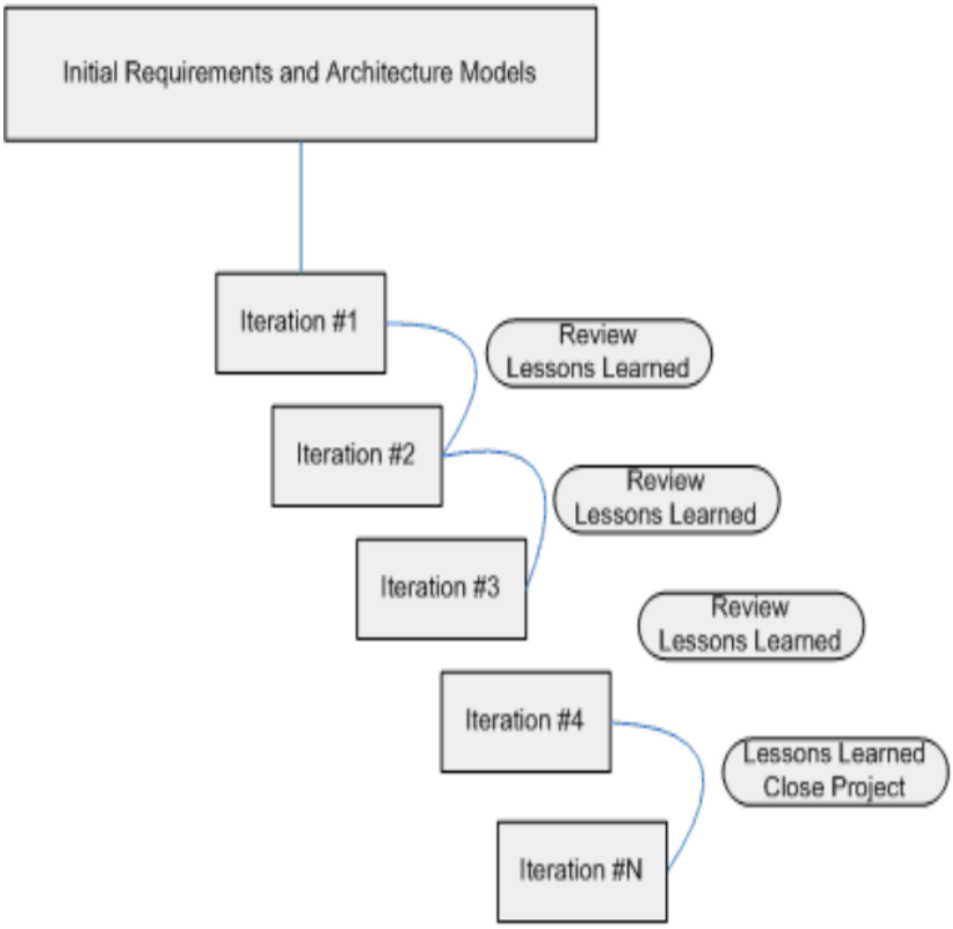


Abbildung 12: Lebenszyklus agiles Modell

(Quelle: Balaji und Murugaiyan, 2012: 29)

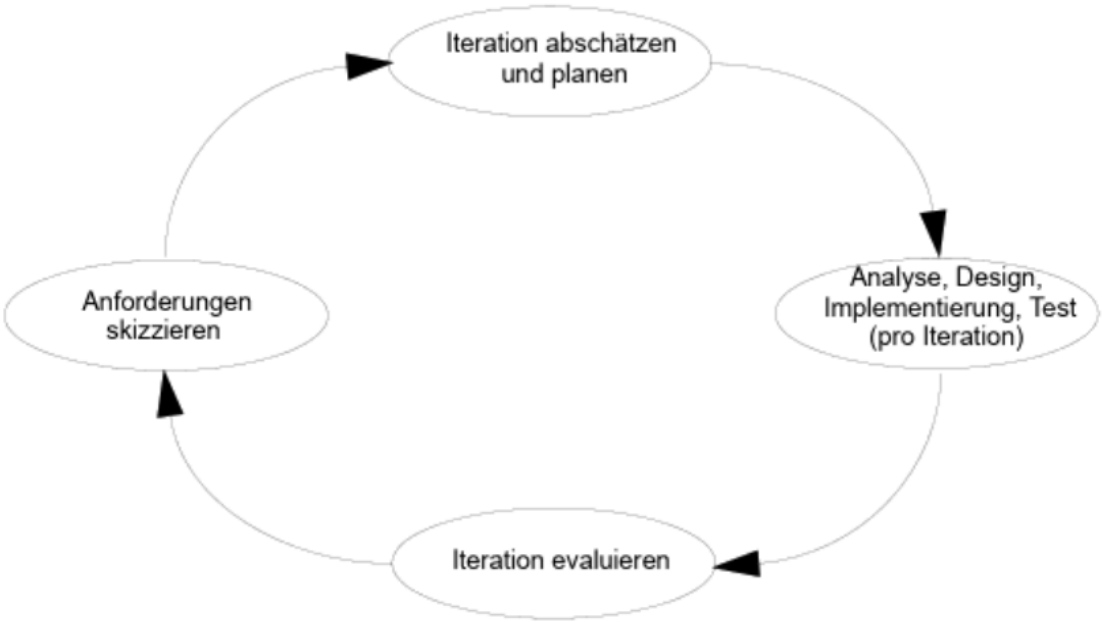


Abbildung 13: Iteration bei einem agilen Ansatz

(Quelle: Bauer und Günzel, 2013: 408).

2.5 Data Warehouse-Automatisierung

„Data-Driven“ ist zum modernen Mantra von Beratern und Anbietern im Datenmanagement geworden. Die Vision von Data Lakes, die mit kostenloser Open Source-Software betrieben werden, verspricht neue Möglichkeiten, um das Kundenverhalten vorherzusagen, die Produktion zu perfektionieren und die Konkurrenz zu verdrängen. Ob aus den sozialen Medien oder dem Internet der Dinge – eine Welle von Daten im Exabyte-Bereich bricht über die Unternehmen herein und hebt ihre Erwartungen an die Daten und ihre Bereitsteller auf ein noch nie dagewesenes Niveau.

Während die Big Data-Welle anrollt, findet in der alten Welt des Data Warehousing eine stille Revolution statt. Neue Ansätze zur Lösung der bisherigen Probleme bei der Entwicklung, Bereitstellung und Wartung von DWHs und Marts sind entstanden. Dank neuer Tools und Techniken wurde damit begonnen, Daten schneller und genauer zu liefern, um die Anforderungen von BI zu erfüllen. Agile Ansätze vereinfachen und beschleunigen die Pflege der Datenressourcen bei sich ändernden Geschäftsanforderungen. Die Rede ist von DWHA, die für Unternehmen, welche sie anwenden, erhebliche Vorteile bringt, siehe Abbildung 14: (vgl. Devlin, 2015).

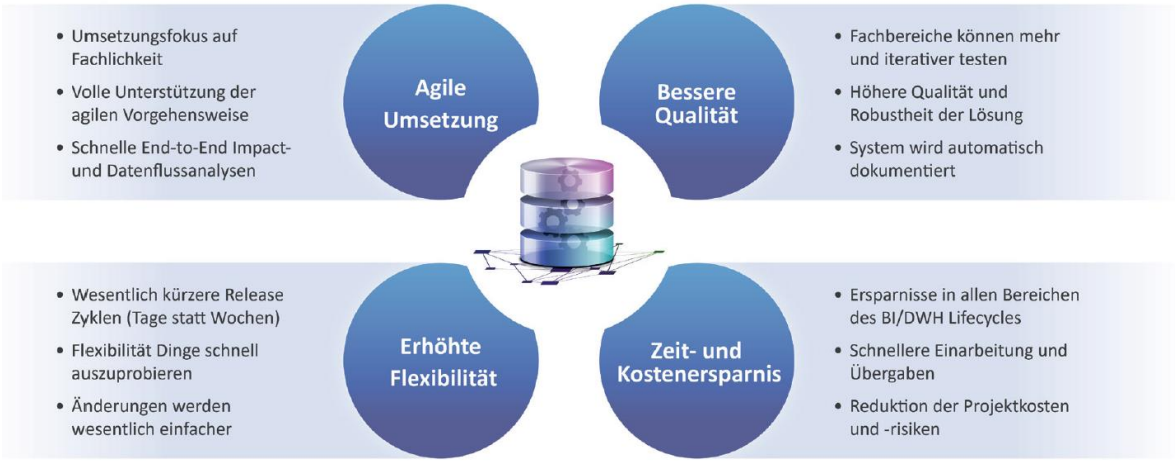


Abbildung 14: Vorteile der Data Warehouse-Automatisierung

(Quelle: Blaschek, 2017:2)

(Anmerkung: Rechtschreibung so im Original und nicht änderbar)

Wie eingangs bereits dargelegt, versteht man unter DWHA den Prozess der Beschleunigung und der Reduktion manueller Vorgänge der DWH-Entwicklungszyklen bei gleichzeitiger Gewährleistung von Qualität und Konsistenz. Bei der klassischen Vorgehensweise formuliert die Fachabteilung ihre Anforderungen im Freitext und gibt sie an die IT-Abteilung beziehungsweise die dafür zuständigen Business-Analysten weiter. Diese spezifizieren und

strukturieren die Anforderungen. Implikationen für die bestehende DWH-Lösung müssen geprüft werden. Daraus lassen sich die Entwicklungsvorgaben für die gewünschte Anpassung des DWH ableiten und für die Umsetzung konkretisieren. Es dauert Wochen oder gar Monate, bis die IT einen Umsetzungsvorschlag unterbreiten kann. Nach Prüfung durch den Fachbereich werden entweder Verbesserungen vorgenommen, oder die bisherige Version wird beibehalten. Die Umsetzung erfolgt anschließend in zeitaufwendiger manueller Entwicklungsarbeit. Mit DWHA ist der gleiche Prozess innerhalb von Stunden oder wenigen Tagen abgeschlossen. Die Release-Zyklen werden durch DWHA verkürzt, und die Basis für eine agile Entwicklungsmethodik wird gelegt. Neue Anforderungen werden in kurzen Iterationen umgesetzt. Bei Bedarf können sie sofort auf einfache Art und Weise angepasst werden. Neben der Tatsache, dass BI-Projekte auf diese Weise mehr als halb so schnell umgesetzt werden können, ist ein weiterer wichtiger Effekt, dass das Verständnis und das Vertrauen der Fachabteilungen für und in die IT wachsen. Abbildung 15 zeigt die Einsparungen, die in den jeweiligen Phasen erzielt werden können. Übertragen auf das gesamte Projekt summieren sich die Einsparungen im Bereich Design und Anforderungsanalyse auf etwa 20 Prozent, und die Reduzierung der Aufwendungen für Entwicklung, Test und Implementierung auf mindestens 50 Prozent oder mehr (vgl. Zeiler, 2017: 2).

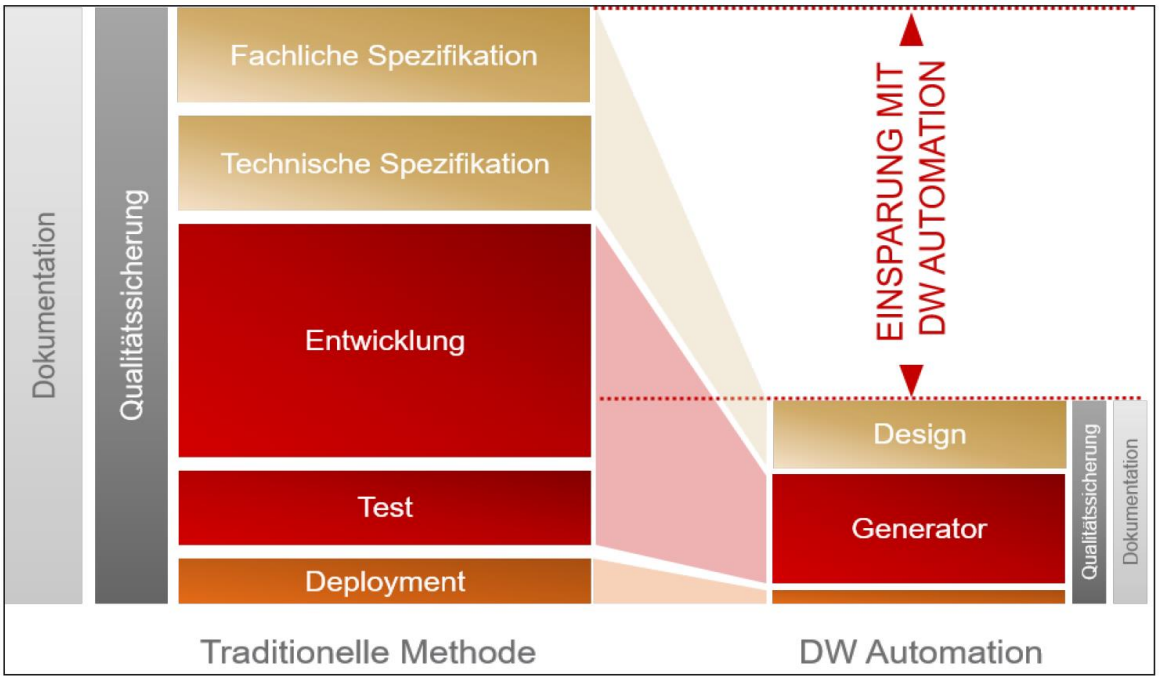


Abbildung 15: Einsparungspotenzial von DWHA

(Quelle: Zeiler, 2017: 2)

(Anmerkung: Rechtschreibung so im Original und nicht änderbar)

Damit DWHA in der Praxis den gewünschten Nutzen bringt und über den gesamten Lebenszyklus einer DWH-Lösung dauerhaft einsatzfähig ist, müssen einige wesentliche Kriterien

beachtet werden. Eine der wichtigsten Eigenschaften eines Werkzeugs für DWHA ist seine Flexibilität. Je mehr ein Tool auf die spezifischen architektonischen Anforderungen des Anwenderunternehmens zugeschnitten und in Bereichen wie Verarbeitungsebenen, Modellierungsoptionen und Namenskonventionen anpassbar ist, desto größer sind der potenzielle Automatisierungsgrad und der daraus resultierende Nutzen. Wenn diese Flexibilität nicht gegeben ist, muss das DWH an die begrenzten Möglichkeiten des Automatisierungstools angepasst werden, da sonst der Automatisierungsumfang leidet, weil nur Teile der DWH-Lösung automatisiert werden können. Abbildung 16 skizziert, was ein High-end-Tool für DWHA bieten sollte: Wenn die dort aufgezeigten Aspekte in ausreichendem Maße vorhanden sind, wird das Wunder der Automatisierung im Data Warehouse wahr (vgl. Zeiler, 2017: 2).

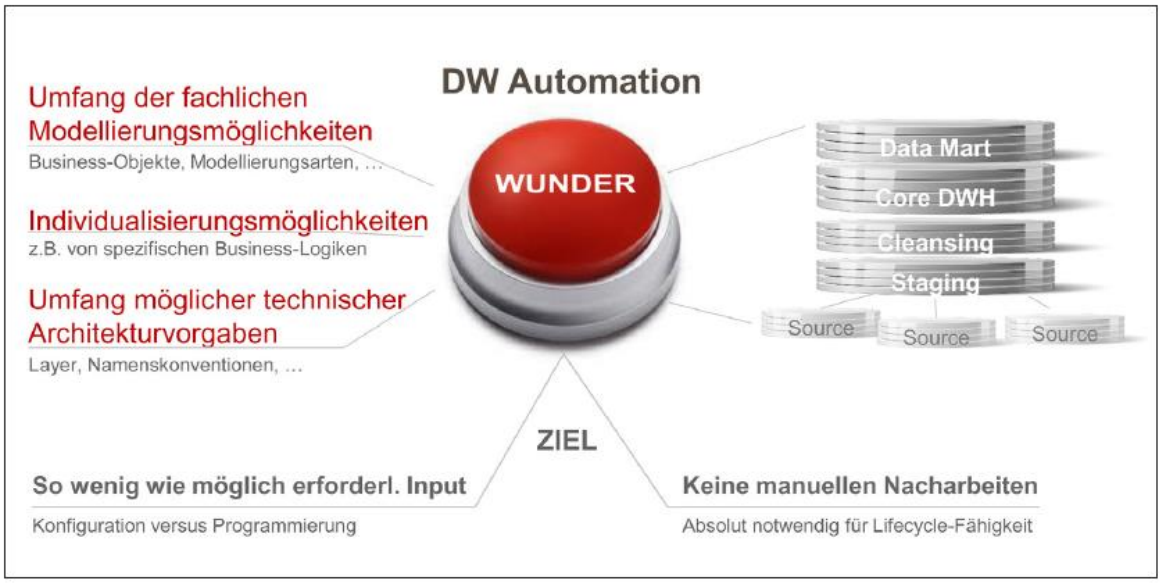


Abbildung 16: Umfang der Automatisierung

(Quelle: Zeiler, 2017: 2)

(Anmerkung: Rechtschreibung so im Original und nicht änderbar).

Kernelement der DWH-Automatisierungswerkzeuge sind Generatoren. Sie erzeugen aus den Metadaten automatisch die gewünschten DWH-Objekte und -Datenflüsse. Entscheidend ist, dass der entsprechende Generator die gewünschte Architektur der DWH-Lösung flexibel unterstützt und die gesamte Geschäftslogik zur Verarbeitung der Daten über das Tool verwaltet. Nur so kann sichergestellt werden, dass die Vorteile der Automatisierung über den gesamten Lebenszyklus der DWH-Lösung ausgeschöpft werden können.

Traditionell ist der Prozess der Anforderungsformulierung über das Requirements Engineering bis hin zum technischen Design langwierig und „verschlingt“ viel Zeit zahlreicher Mitarbeiter. Bei der Automatisierung hingegen werden aus Metadaten Skripte generiert, mit denen die gesamte DWH-Lösung erstellt wird. Der Schlüssel zur Automatisierung sind also die

beschreibenden Metadaten. Sie müssen in einer sauberen Struktur für die Verarbeitung im Generator abgelegt werden. Damit der volle Umfang von DWHA in der Entwurfsphase genutzt werden kann, ist eine Unterstützung für die strukturierte Erfassung dieser Metadaten erforderlich. In diesem Zusammenhang kommen die Unterschiede zum traditionellen Ansatz mit einem sehr aufwendigen Prozess der Spezifikation und des Designs zum Tragen. Während in der Regel mehrere Spezifikationsschritte in unterschiedlichen Genauigkeitsstufen notwendig sind, liegt der Fokus in der Entwurfsphase mit Automatisierungswerkzeugen auf dem gezielten Aufbau der Metadaten. Werkzeuggestützte Validierung hilft hier, lückenlos und konsistent vorzugehen. Dadurch wird die Designphase auch wesentlich effizienter als bei der traditionellen Vorgehensweise (vgl. Zeiler, 2017: 3).

DWHA trägt Data Warehousing in die Zukunft. Für IT-Abteilungen ist sie der Schlüssel, um BI/DWH-Projekte schnell, flexibel und effizient umzusetzen. Für die Fachabteilungen ist sie eine zentrale Unterstützung ihrer geschäftskritischen Aktivitäten. Für das Unternehmen als Ganzes ist die Automatisierung eine Quelle für neue Geschäfts- und Produktstrategien (vgl. Zeiler, 2017: 4).

2.5.1 Facetten der BI/DWH-Automatisierung

Um die geforderten Effizienz- und Qualitätssteigerungen im Bereich BI/DWH zu erreichen, ist der Einsatz von Automatisierungsmethoden und -techniken unumgänglich. Der BI/DWH-Automatisierungsansatz betrifft nicht nur die Entwicklung, sondern gleichermaßen auch die Infrastruktur, das Testing, das Deployment, den Betrieb und die Dokumentation; auf ihre jeweiligen Eigenschaften wird im Folgenden genauer eingegangen (vgl. Blaschek, 2017: 1 f.):

- Infrastruktur

  Die Nutzung der Vorteile des Automatisierungsansatzes beginnt mit der Auswahl der BI/DWH-Infrastruktur. Die stark steigende Nachfrage von Unternehmen nach BI/DWH-Lösungen in der Cloud und das kontinuierliche Wachstum der Anzahl von BI/DWH-PaaS(Platform-as-a-Service)- und BI/DWH-SaaS(Software-as-a-Service)-Lösungen zeigen, dass die Automatisierung der Infrastruktur den Unternehmen nachweisbare wirtschaftliche, organisatorische und architektonische Wettbewerbsvorteile durch Skalierung, Backup, Fail-Over, Update, Überwachung etc. bietet.
- Entwicklung

  Auch bei der individuellen Entwicklung von BI/DWH-Lösungen können Unternehmen vom Einsatz der Automatisierung profitieren. Während die weit verbreiteten ETL/ELT-Entwicklungswerkzeuge schon immer Datenbankcode aus Metadaten

generiert haben, bieten moderne DWH-Werkzeuge, spezialisiert auf den Automatisierungsansatz, deutlich mehr Möglichkeiten: Im Gegensatz zu klassischen Werkzeugen können diese alle Metadaten zentral generieren und modifizieren, einschließlich der Datenbank- und Job-Objekte. Dadurch werden Änderungen nicht nur viel schneller durchgängig, sondern auch deutlich robuster und qualitativ hochwertiger umgesetzt.

- Testing

  Der Einsatz von Automatisierungsmethodik unterstützt auch das Testen von BI/DWH-Lösungen. Beispiele im Bereich ‚Testen' sind die automatische Generierung von Testdaten, die automatische Ausführung von Regressionstests nach dem Ansatz der testgetriebenen Entwicklung und kontinuierlichen Integration, die Test-automatisierung von Reports und/oder die automatische Datenqualitätssicherung, bei der definierte Testfälle mit Ausgabe- und Zielergebnissen in einer Testdatenbank gespeichert werden.

- Deployment

  Der Automatisierungsansatz bringt zudem Vorteile beim Deployment von BI/DWH-Objekten. So kann beispielsweise die optimale Ausführungsreihenfolge von Transformationsjobs (Scheduling) automatisch aus den Metadaten generiert werden, inklusive Logging und Fehlerbehandlung. Der Build- und Release-Prozess, der sehr zeitaufwendig und fehleranfällig ist, insbesondere dann, wenn Änderungen an bestehenden DWH-Strukturen vorgenommen werden, kann durch die Verwendung von Metadaten ebenfalls automatisiert werden. Beschleunigte und stabilere Deployments machen agiles Vorgehen mit kurzen Sprintzeiten im BI/DWH-Bereich erst möglich.

- Betrieb

  Die Verwendung von Automatisierung in BI/DWH-Betriebsthemen kann ebenfalls spürbare Optimierungen erzielen. Neben den oben beschriebenen Möglichkeiten in den Bereichen Logging, Fehlerbehandlung und Scheduling können automatisierte Incident-Prozesse (z. B. die automatische Erstellung von Tickets mit angereicherten Meta- und Laufzeitdaten) den Analyse- und Wartungsaufwand deutlich reduzieren. Darüber hinaus helfen automatisierte Prozesse in den Bereichen Benutzer- und Berechtigungsmanagement oder Object-Lifecycle-Management, die BI/DWH-Lösung schneller und effizienter zu betreiben.

- Dokumentation

  Ein oft verkannter, aber immenser Vorteil, der sich aus der konsequenten Nutzung von BI/DWH-Metadaten ergibt, ist die Automatisierbarkeit der BI/DWH-Dokumentation, die es gestattet, einen Großteil der Dokumentationsarbeit auf „Knopfdruck“ zu erstellen. So können Datenfluss- und Auswirkungsanalysen im Gegensatz zu manuell dokumentierten BI/DWH-Systemen in Minuten statt in Tagen durchgeführt werden.

2.5.2 Data Warehouse-Automatisierungstools: Designansätze

Bei der DWHA gibt es ebenfalls ein breites Spektrum an Lösungen. Zwei Hauptansätze können unterschieden werden (vgl. Eckerson, 2015: 6-8; und Branger, 2017:1):

1. Model-Driven:

   Bei diesem Ansatz wird versucht, die Anforderungen zunächst in einem konzeptionellen und oder logischen Datenmodell zu erfassen und anschließend zu implementieren. Die Umsetzung in physische Modelle und Code wird durch die logischen Modelle gesteuert. Im Falle eines Fehlers (z. B. durch unerwartete Eigenheiten in den Quelldaten) muss das Modell zunächst angepasst, und die Implementierung muss neu generiert werden.
2. Data-Driven:

   Bei diesem Ansatz wird so früh wie möglich direkt mit den vorhandenen Quelldaten gearbeitet und das DWH schrittweise auf der physikalischen Schicht implementiert. Darauf aufbauend kann die Implementierung angepasst werden.

2.5.3 Data Warehouse-Automatisierungstools: Motive für den Einstieg

Der Einstieg in ein DWHA erfolgt in der Regel aus wirtschaftlichen Gründen. Die fünf häufigsten Einstiegsszenarien sind nachfolgend skizziert (vgl. Zeiler, 2017: 3 f.):

- Neue Anforderungen zeitnah umsetzen

  Aufgrund neuer Anforderungen von innen, beispielsweise durch Fachbereiche, oder von außen, wie z. B. durch Gesetze und Vorschriften, muss die bestehende DWH-Lösung schnell erweitert werden. Es wird alsbald deutlich, dass dies mit dem bestehenden DWH nicht möglich ist. Eine gute Option besteht darin, ein zusätzliches DWH einzuführen und einen Parallelbetrieb zum bestehenden System mit DWHA zu fahren. Die Daten sind in diesem Fall nicht redundant. Nach und nach werden immer mehr Projekte auf dem neuen System realisiert, bis alles komplett darauf umgestellt ist.

Dieser Parallelbetrieb kann zwar einige Monate dauern, hat aber keinen Einfluss auf die Produktivität.

- Werkzeuge oder Plattformen werden abgekündigt

  Wenn Anbieter von ETL-Werkzeugen oder -Plattformen nicht mehr unterstützt werden, wenn die Lizenzbedingungen nicht mehr akzeptabel sind oder wenn Daten in die Cloud migriert werden sollen, steht ein Wechsel zu neuen Technologien und Werkzeugen an. Dabei werden die bestehenden Strukturen und Geschäftslogiken auf das neue System übertragen. Das heißt, dass die notwendigen ETL-Skripte für die Migration in diesem Fall von einem DWHA-Tool automatisch generiert werden, nachdem die Modelldaten einmal gepflegt wurden. Die kurzen Release-Zyklen ermöglichen ein agiles und iteratives Vorgehen im Migrationsprojekt.

- Strategische Modernisierung des DWH

  Wenn der Aufwand für Betrieb und Wartung der bestehenden DWH-Lösung nicht mehr vertretbar ist und neue Anforderungen nur sehr zeit- und kostenintensiv umgesetzt werden können, rücken DWH-Automatisierungswerkzeuge in den Fokus des Interesses und der Überlegungen. Die Handlungsoptionen reichen von einer sanften Migration bis hin zu einem Greenfield-Ansatz[2].

- Integration von Datensilos

  Datensilos haben sich historisch als Insellösungen in einzelnen Abteilungen entwickelt. Sie verhindern oft umfassende Auswertungen des gesamten Datensatzes. Die einzelnen Auswertungsstrukturen sind nicht homogenisiert, und redundante Datenspeicher machen die bestehende Situation noch schwieriger. Der Aufwand zur Pflege von Datensilos ist ebenso groß wie unnötig. Hier erlauben einige DWHA-Ansätze, bestehende Datensilos zu integrieren und als gemeinsame Objekte in einem gemeinsamen Hub zu verwalten. Die Pflege und Verwaltung der Metadaten findet dann an einer zentralen Stelle statt. Dadurch ist es möglich, bereichsübergreifende Beziehungen ohne Redundanz zu erstellen.

- Compliance-Anforderungen erfüllen

  Viele Vorschriften wie DSGVO, BCBS 239 und Compliance-Regelungen erfordern eine exakte Nachvollziehbarkeit der Daten und eine saubere Dokumentation der Datenverarbeitungsprozesse. Alle Datenbereinigungen und -korrekturen müssen nachvollziehbar dokumentiert werden. Dies gilt auch für die Verarbeitung von

[2] Bei diesem Ansatz wird ein Projekt von Grund auf, wie auf einer – bildlich gesprochen – freien, grünen Wiese, eigenständig und ohne Schnittstelle zum bestehenden Unternehmen entwickelt (vgl. Fiore, 2018).

Schnittstellen. Werden die Compliance-Anforderungen nicht oder nur teilweise erfüllt, kann dies zu schlechten Ratings, negativen Risikobewertungen und/oder sogar zu Strafen führen. DWHA-Tools pflegen die Metadaten eines DWH und generieren die Aufbereitungsprozesse entsprechend. Auf diese Weise werden die Prozesse standardisiert und implementiert, aber auch automatisch dokumentiert. Änderungen an Datenstrukturen und Aufbereitungsprozessen sind vergleichbar und können nachvollzogen werden. Dies erleichtert die Umsetzung von Compliance-Anforderungen und reduziert das Risiko, gegen Richtlinien zu verstoßen.

3 Verwendete Technologien und Werkzeuge

Die zur Gegenüberstellung der Performance- und Nutzenanalyse verwendeten Technologien und Werkzeuge werden im Folgenden näher beschrieben. Auf die Beschreibung allgemein bekannter Programme und Hilfsmittel (z. B. Microsoft Excel) wird verzichtet.

3.1 Microsoft Azure

Die Azure-Plattform wurde auf der Basis des bekannten und ausgiebig getesteten SQL-Server-Frameworks aufgebaut (vgl. Microsoft, 2017: 4). Die Kombinierbarkeit von entsprechender Komponenten eröffnet die Möglichkeit, ein System in der Cloud aufzubauen, das starke Ähnlichkeiten mit bekannten Architekturen und Softwarekomponenten aufweist, aber auch um beliebige neue Elemente erweitert werden kann. Auf diese Weise sind vertraute Abläufe ebenso wie die vielfältigen Möglichkeiten und die Flexibilität der Cloud gleichzeitig gewährleistet. Die Konsistenz von On-Premises- und Cloud-Anwendungen unterstützt auch den Ansatz einer hybriden Architektur (vgl. Microsoft, 2017: 4), da cloudbasierte Anwendungen mit lokalen Softwarekomponenten kombiniert werden und mit diesen interagieren können. Die Komponenten passen sich dank der Skalierbarkeit ganz an ihren Einsatzzweck an und können sowohl von kleinen und mittleren Unternehmen für weniger umfangreiche Anwendungsfälle genutzt als auch zu Hochleistungssystemen skaliert werden. Im Folgenden werden die beiden Azure-Komponenten betrachtet, die für die Performance-Nutzen-Analyse verwendet wurden. Ausgenommen ist hierbei die Azure SQL-Datenbank, die in Abschnitt 3.4.1 separat beschrieben wird.

- Azure Virtual Machine (AVM)

  Bei der Azure Virtual Machine handelt es sich um eine von mehreren bedarfsgesteuerten, skalierbaren Computerressourcen. Die AVM bietet die Flexibilität der Virtualisierung, jedoch ohne die Zeit und das Geld für den Kauf und die Verwaltung der Hardware zu beanspruchen (vgl. Virtuelle Linux-Computer in Azure, in: Microsoft [Stichwort], 2019).
- Azure Files

  Bei den Azure Files handelt es sich um eine vollständig verwaltete Dateifreigabe in der Cloud, auf die über Branchenstandardprotokolle zugegriffen werden kann. Auf Azure Files kann gleichzeitig über die Cloud und durch eine eingebundene Bereitstellungen zugegriffen werden (vgl. Was ist Azure Files?, in: Microsoft [Stichwort], 2020).

3.2 Docker

Docker ist eine offene Plattform, die für die Entwicklung, Auslieferung und Ausführung von Anwendungen verwendet werden kann. Mit Docker können Anwendungen in einer isolierten Umgebung verpackt und ausgeführt werden. Um eine ordnungsgemäße Ausführung der Anwendung zu ermöglichen, werden alle dafür benötigten Anforderungen in Form von Bibliotheken oder Binärdateien in den entsprechenden Container gepackt (vgl. Docker overview, in: Docker [Stichwort], o. J.). Dieser Container enthält somit alles, was die Anwendung zur Ausführung benötigt. Die Isolierung von Anwendungen durch Container ermöglicht es, verschiedene Container auf beliebigen Hosts zu betreiben und das darunterliegende Betriebssystem zu vernachlässigen. Weiterhin werden neben dem zugrundeliegenden Betriebssystem auch alle anderen Abhängigkeiten innerhalb des Containers definiert, sodass keine weiteren Installationen auf dem Hostsystem vorgenommen werden müssen. Um eine solche isolierte und sichere Umgebung für die Entwicklung und Ausführung einer Anwendung aufzubauen, werden bestimmte Docker-Objekte, wie Images, Container oder Volumes, verwendet. Um eine Vorstellung von der Funktionsweise von Docker zu vermitteln, werden im Folgenden die einzelnen Objekte genauer vorgestellt.

3.2.1 Images

Um einen Docker-Container zu erstellen, benötigt Docker einen Befehlssatz von Anweisungen. Diese Anweisungen werden in sogenannten Images niedergeschrieben. Um ein solches Image zu erstellen, muss ein Dockerfile geschrieben werden. Dieses Dockerfile enthält dann alle Anweisungen, die benötigt werden, um das gewünschte Image zu erstellen. Jede dieser Anweisungen stellt eine sogenannte Schicht innerhalb des Images dar. Diese Schichten bauen aufeinander auf. Dadurch ist es möglich, ein bestehendes Docker-Image zu erweitern und weitere Schichten hinzuzufügen, um die gewünschte Umgebung für die Anwendung zu schaffen. Neben der Erstellung eigener Images ist es auch möglich, bestehende Images herunterzuladen und zu verwenden. Diese Images werden in einem Repository „Docker Hub“ persistiert und sind öffentlich zugänglich sowie wiederverwendbar. Durch die einfache Erweiterbarkeit eines Images kann z. B. ein Ubuntu-Image, das ein Linux-Betriebssystem virtualisiert, heruntergeladen und um eine Python-Installation ergänzt werden (vgl. Docker overview, in: Docker [Stichwort], o. J.).

3.2.2 Container

Ein Container ist die lauffähige Instanz eines zuvor beschriebenen Images. Anhand der im Dockerfile geschriebenen Anweisungen wird Schritt für Schritt die gewünschte Umgebung mit allen notwendigen Voraussetzungen und Abhängigkeiten installiert. Die Container können vom Benutzer nach Belieben erstellt, gestartet, gestoppt oder gelöscht werden. Standardmäßig ist der Container von seinem Host-System isoliert. Der Benutzer kann jedoch definieren, ob bestimmte Abhängigkeiten zwischen dem Host-System und dem Container erlaubt sein sollen. So kann z. B. definiert werden, ob bestimmte Daten auf dem Host-System persistiert werden sollen oder ob der Container über das Host-System oder darüber hinaus auf dem Netzwerk sichtbar oder zugänglich sein soll (vgl. Docker overview, in: Docker [Stichwort], o. J.).

3.2.3 Volumes

Wie bereits erwähnt, werden containerisierte Anwendungen über Docker vom Host-System isoliert. Daten sind daher nur über den Lebenszyklus des Containers verfügbar und werden beim Beenden oder Neustart des Containers gelöscht. Möchte man jedoch Daten über einen Ausführungszeitraum hinaus persistieren oder Daten des Host-Systems innerhalb des Docker-Containers nutzen, erstellen, manipulieren oder löschen, müssen konkrete Abhängigkeiten zwischen den Containern und dem Host-System definiert werden. Zu diesem Zweck

werden sogenannte Volumes verwendet. Mit Volumes wird definiert, welche Ordner und Dateien vom Host-System und dem Container gemeinsam genutzt werden. Alle Änderungen, die im Container oder im Host-System vorgenommen werden, wirken sich also direkt auf die jeweils andere Komponente aus (vgl. Docker overview, in: Docker [Stichwort], o. J.).

3.3 Docker Compose

Docker Compose ist ein Werkzeug zur Definition und Ausführung von Anwendungen über mehrere Container hinweg. Die einzelnen Dienste der Anwendung werden in einer YAML-Datei konfiguriert. Jeder der in dieser docker-compose.yml beschriebenen Dienste greift auf das in den jeweiligen Dockerfiles beschriebene Image zu und baut so die einzelnen Container auf. Jeder dieser Container basiert auf einem eigenen Dockerfile und kann auch ohne Docker Compose, wie zuvor dargelegt, einzeln gestartet werden. Der Vorteil von Docker Compose besteht allerdings darin, dass die einzelnen Dienste, die für die gesamte Anwendung benötigt werden, in einer einzigen Datei beschrieben und konfiguriert werden können. Dabei hat man auch die Möglichkeit, Abhängigkeiten zwischen den Diensten zu definieren. Neben den Abhängigkeiten zwischen den Diensten können auch Volumes (siehe Abschnitt 3.2.3) definiert oder Ports freigegeben werden, auf die der Anwender dann über die Host-Adresse zugreifen kann. Sind alle Dienste mit ihrer Konfiguration und ihren Abhängigkeiten in der Datei docker-compose.yml definiert, kann die gesamte Anwendung oder die daraus bestehenden Dienste über den Befehl ‘docker-compose up’ gestartet werden (vgl. Use Docker Compose, in: Docker [Stichwort], o. J.).

3.4 Datavault Builder

Der Datavault Builder (DVB) der 2150 GmbH stellt eine professionelle Lösung für die Implementierung einer agilen DWH-Architektur dar. Er unterstützt die Aufgabe mit modernsten Modellierungstechniken und einem geführten Workflow. Dabei deckt das Tool alle Phasen und Funktionen der flexiblen DWH-Entwicklung und des Betriebs auf Basis der Data Vault-Modellierung ab.

Der DVB erstellt durch Ausführen eines Skripts eine Data Vault-Architektur innerhalb der Zieldatenbank inklusive der notwendigen ETL/ELT-Prozesse. Das modellierte logische Design wird anschließend in eine funktionierende, technische Implementierung umgesetzt. Die direkte Anbindung an unterschiedliche Quellsysteme wird durch verschiedene Konnektoren

sichergestellt (vgl. Datavault Builder – Ihr Data Warehouse Automatisierungs-Werkzeug der 4. Generation, in: Xax [Stichwort], o. J.)

Die Bereitstellung des DVB erfolgt über Docker-Container. Damit ist das Tool infrastruktur- und betriebssystemunabhängig und kann überall eingesetzt werden, sowohl On-Premises als auch in der Cloud. Testumgebungen auf Desktop-Rechnern sind ebenfalls möglich.

Der Einsatz der Data Vault-Modellierungsmethodik stellt sicher, dass das Datenmodell im Verlauf des Projekts problemlos erweitert werden kann. Neue Daten können in das bestehende Modell integriert werden, ohne dass zeitaufwendige Änderungen an bestehenden Objekten vorgenommen werden müssen.

Durch die Entkopplung der bestehenden Datenobjekte beziehungsweise Entitäten im Modell ermöglicht die Data Vault-Methode ein hochparalleles Laden und damit auch eine effiziente Weiterverarbeitung der Daten bis in die Reporting-Schicht des DWH (vgl. Visuelle Datawarehouse Entwicklung – effizienter geht nicht, in: Ventum-consulting [Stichwort], o. J.)

3.5 Datenbankserver (Zielplattformen)

Bei Verwendung des DVB besteht die Möglichkeit, eine Verbindung zu einer externen Zieldatenbank herzustellen. Die in der Performance-Nutzen-Analyse untersuchten Zieldatenbanken ‚Azure SQL-Datenbank', ‚Exasol' und ‚Snowflake' werden im Folgenden vorgestellt.

3.5.1 Azure SQL-Datenbank

Die Azure SQL-Datenbank ist eine vollständig verwaltete PaaS-Datenbank-Engine, welche die meisten Datenbankverwaltungsfunktionen ohne Benutzereingriff durchführt, wie z. B. Upgrades, Patches, Backups und Überwachung. Die Azure SQL-Datenbank führt immer die neueste, stabile Version der SQL-Server-Datenbank-Engine aus und läuft auf einem gepatchten Betriebssystem mit 99,99 % Verfügbarkeit. Die in der Azure SQL-Datenbank integrierten PaaS-Funktionen ermöglichen es, sich ausschließlich auf domänenspezifische Datenbankmanagement- und Optimierungsaktivitäten zu konzentrieren.

Die Azure SQL-Datenbank kann dazu verwendet werden, um hochverfügbare und hochleistungsfähige Datenspeicherebenen für Anwendungen und Lösungen in Azure zu erstellen. Sie ermöglicht es, sowohl relationale Daten als auch nicht-relationale Strukturen, wie z. B. Graph-JSON-, Spatial- und XML-Daten zu verarbeiten.

Die Azure SQL-Datenbank ist ein vollständig verwalteter Dienst mit integrierter Hochverfügbarkeit, Backup und anderen üblichen Wartungsvorgängen. Microsoft übernimmt das Patchen und Aktualisieren des SQL- und Betriebssystemcodes. Es muss also anwenderseitig keine Wartung der zugrunde liegende Infrastruktur vorgenommen werden.

Die Azure SQL-Datenbank bietet die folgenden Datenbankbereitstellungsoptionen:

- Einzeldatenbank (auch Singleton genannt) stellt eine vollständig verwaltete, isolierte Datenbank dar. Diese Option kann gewählt werden, wenn moderne Cloud-Anwendungen und Microservices bereitgestellt werden, die eine zentralisierte, zuverlässige Datenquelle benötigen. Ein Singleton ist mit einer eigenständigen Datenbank in der SQL-Server-Datenbank-Engine zu vergleichen.
- Ein elastischer Datenbankpool ist eine Sammlung von Singleton-Datenbanken mit gemeinsam genutzten Ressourcen wie CPU oder Speicher. Singletons können in und aus elastische/n Datenbankpools verschoben werden.

Die Azure SQL-Datenbank bietet eine vorhersehbare Leistung mit mehreren Ressourcentypen, Service-Tiers und Rechengrößen. Diese Lösung bietet dynamische Skalierbarkeit ohne Ausfallzeiten, integrierte, intelligente Optimierung, globale Skalierbarkeit und Verfügbarkeit sowie erweiterte Sicherheitsoptionen. Mit diesen Funktionen können sich Anwender auf die schnelle App-Entwicklung und die Verkürzung der Time-to-Market konzentrieren, statt sich mit der Verwaltung virtueller Maschinen und der Infrastruktur zu befassen. Die Azure SQL-Datenbank wird derzeit weltweit in 38 Rechenzentren betrieben, sodass die Datenbank in einem Rechenzentrum in der Nähe des Anwenders laufen kann.

Die Menge der zugewiesenen Ressourcen kann wie folgt festgelegt werden:

- Bei einzelnen Datenbanken ist jede Datenbank von den anderen isoliert und portabel. Jede hat ihre eigene garantierte Menge an Compute-, Arbeitsspeicher- und Speicherressourcen. Die Menge der der Datenbank zugewiesenen Ressourcen ist für diese Datenbank eindeutig und wird nicht mit anderen Datenbanken in Azure geteilt. Die Ressourcen einer einzelnen Datenbank können dynamisch nach oben und unten skaliert werden. Die Einzeldatenbankoption weist unterschiedliche Compute-, Arbeitsspeicher- und Speicherressourcen für unterschiedliche Anforderungen von Anwendungen zu. So können diesen beispielsweise 1 bis 80 virtuelle Kerne oder 32 GB bis 4 TB Speicher zugewiesen werden. Mit der Hyperscale-Serviceschicht für Singletons können bis zu 100 TB mit schnellen Backup- und Restore-Funktionen skaliert werden.

- Mit Pools für elastische Datenbanken lassen sich Ressourcen zuweisen, die von allen Datenbanken im Pool gemeinsam genutzt werden. Es kann entweder eine neue Datenbank erstellt oder bestehende einzelne Datenbanken in einen Ressourcenpool verschoben werden, um die Ressourcenauslastung zu maximieren und auf diese Weise Kosten zu reduzieren. Diese Option bietet auch die Möglichkeit, die Ressourcen im Pool für elastische Datenbanken dynamisch nach oben und unten zu skalieren.

Die erste App für eine kleine Einzeldatenbank kann im Service-Tarif „Universal“ zu geringen monatlichen Kosten erstellt werden. Bei Bedarf kann der Service-Tarif jederzeit manuell oder programmatisch auf „unternehmenskritisch“ geändert werden, um den Anforderungen der Lösung gerecht zu werden. Performance-Tuning ist möglich, ohne Ausfallzeiten für die App oder den Kunden zu verursachen. Durch die dynamische Skalierung kann die Datenbank transparent auf sich schnell ändernde Ressourcenanforderungen reagieren. Es werden nur diejenigen Ressourcen bezahlt, die in dem jeweiligen Moment benötigt werden.

Zu beachten ist, dass dynamische Skalierbarkeit nicht dasselbe ist wie automatische Skalierung. Bei dieser wird ein Dienst automatisch anhand von Kriterien skaliert. Die dynamische Skalierbarkeit hingegen ermöglicht eine manuelle Skalierung ohne Ausfallzeit. Die Einzeldatenbankoption unterstützt manuelle dynamische Skalierbarkeit, aber keine automatische Skalierung. Ein höherer Automatisierungsgrad kann bei Bedarf mit elastischen Datenbankpools erreicht werden, welche die gemeinsame Nutzung eines Ressourcenpools, basierend auf den individuellen Datenbankanforderungen, ermöglichen. Eine weitere Option sind Skripte, welche die Automatisierung der Skalierbarkeit einer einzelnen Datenbank unterstützen.

Die folgenden Kaufmodelle sind für die Azure SQL-Datenbank verfügbar:

- Beim vCore-basierten Kaufmodell kann die Anzahl der virtuellen Kerne, die Menge des Speichers sowie die Menge und Geschwindigkeit des Speichers ausgewählt werden. Mit dem vCore-basierten Kaufmodell kann auch der Azure-Hybrid-Vorteil für SQL-Server genutzt werden, um Kosten zu sparen.
- Das DTU-basierte Kaufmodell bietet eine Mischung aus Compute-, Speicher- und I/O-Ressourcen auf drei Service-Tiers zur Unterstützung einfacher bis komplexer Datenbank-Workloads. Die Compute-Größen der einzelnen Tiers bieten unterschiedliche Ressourcen-Mixe, zu denen zusätzliche Speicherressourcen hinzugefügt werden können.
- Beim serverless-basierten Kaufmodell wird die Rechenkapazität je nach Workload-Bedarf skaliert, und die genutzten Rechenressourcen werden auf Basis der Menge pro

Sekunde abgerechnet. Im Serverless Computing-Tarif werden Datenbanken während inaktiver Zeiträume auch automatisch pausiert, wenn nur Speicher abgerechnet wird, und automatisch neu gestartet, wenn die Aktivität wieder aufgenommen wird.

Die Azure SQL-Datenbank bietet drei Serviceebenen, die für verschiedene Arten von Anwendungen konzipiert sind:

1. die Universal/Standard-Servicestufe, die für gängige Workloads konzipiert ist. Diese Ebene bietet budgetorientierte, ausgewogene Rechen- und Speicheroptionen;
2. die Enterprise Critical/Premium-Servicestufe, die für OLTP-Anwendungen mit hohen Transaktionsraten und I/O-Vorgängen mit niedrigster Latenz ausgelegt ist. Dieser Tier bietet dank mehrerer isolierter Replikate die höchste Ausfallsicherheit;
3. die Hyperscale-Servicestufe, die für sehr große OLTP-Datenbanken mit der Funktion zur automatischen Skalierung des Speichers und nahtlosen Skalierung der Rechenressourcen konzipiert ist (vgl. Was ist Azure SQL-Datenbank?, in: Microsoft [Stichwort], 2020).

3.5.2 Exasol

Exasol bietet eine leistungsstarke und hoch performante In-Memory-Analysedatenbank. Ihre Leistung, Skalierbarkeit und Benutzerfreundlichkeit ermöglichen die Analyse von Milliarden von Zeilen in Sekunden, die sichere Ausführung von Hochleistungsanalysen in der Cloud oder On-Premises und die Durchführung von Analysen mit Selbstindizierung, die die Leistung automatisch optimiert (vgl. Exasol – die hoch performante In-Memory-Analytics-Datenbank, in: Areto [Stichwort], o. J.). Abbildung 17 gibt einen Überblick über Exasol und seine Funktionen:

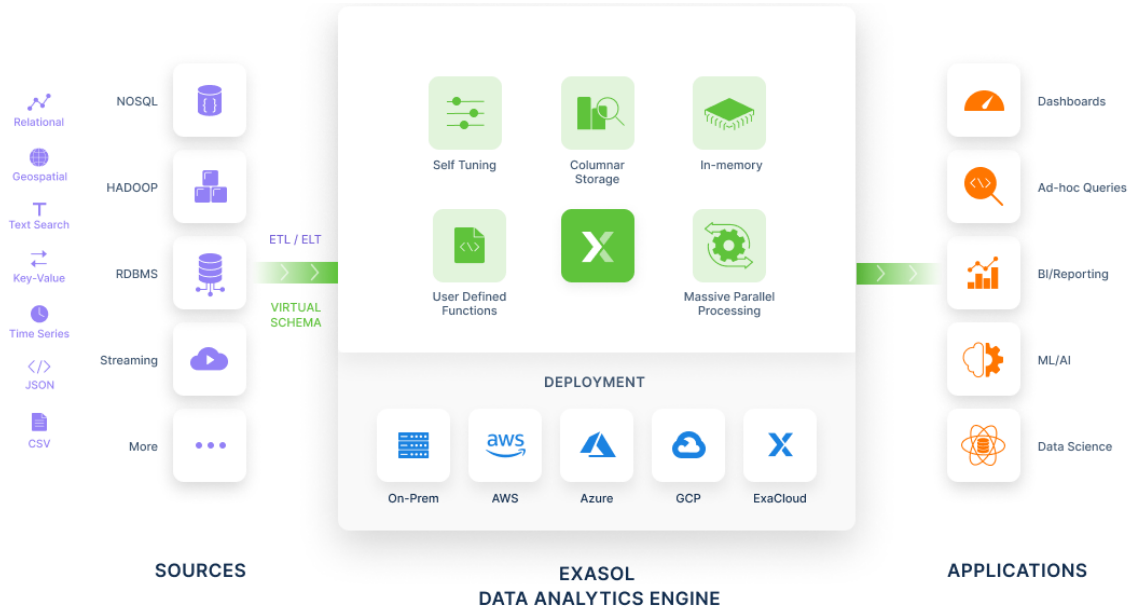


Abbildung 17: Überblick über Exasol und seine Funktionen

(Quelle: Exasol Overview, in: Exasol [Stichwort], o. J.)

Die Exasol Datenbank umfasst folgende Hauptmerkmale (vgl. Exasol Overview, in: Exasol [Stichwort], o. J.):

- In-Memory-Technologie: In-Memory-Algorithmen ermöglichen die Verarbeitung großer Datenmengen im Hauptspeicher und sorgen für schnelle Zugriffszeiten.
- Spaltenbasierte Speicherung und Komprimierung: Die spaltenbasierte Speicherung und Komprimierung reduzieren die Anzahl der I/O-Operationen und die Datenmenge, die für die Verarbeitung im Hauptspeicher benötigt wird, und beschleunigt die analytische Leistung.
- Massiv parallele Verarbeitung (MPP): Exasol wurde als paralleles System entwickelt, das auf einer Shared Nothing-Architektur basiert. Abfragen werden über alle Knoten in einem Cluster verteilt, wobei optimierte, parallele Algorithmen verwendet werden, welche Daten lokal im Hauptspeicher jedes Knotens verarbeiten.
- Hohe Benutzerparallelität: Tausende von Benutzern können gleichzeitig auf große Datenmengen zugreifen und diese analysieren, ohne dass die Abfrageleistung beeinträchtigt wird.
- Skalierbarkeit: Dank der linearen Skalierbarkeit können Systeme erweitert und die Leistung durch Hinzufügen weiterer Knoten gesteigert werden.
- Selbstoptimierung: Intelligente Algorithmen überwachen die Nutzung und führen automatisch Selbstoptimierungsaufgaben durch, die die Leistung optimieren und den Aufwand für die Datenverwaltung minimieren.

- Umfassende Unterstützung für Hadoop: Exasol unterstützt alle nativen HDFS-Formate, sodass schnelle und einfache High-Speed-Analysen mit strukturierten und unstrukturierten Daten durchgeführt werden können.
- Schneller Zugriff auf mehr Datenquellen: Durch ein Datenvirtualisierungs-Framework, genannt „virtual schemas“, sowie ein hochleistungsfähiges Datenintegrations-Framework lassen sich Daten aus mehr Quellen als je zuvor verbinden und analysieren.
- Erweiterte datenbankinterne Analysen: Zusätzlich zu der Out-of-the-Box-Unterstützung für R, Python, Java und Lua lässt sich mit Exasol eine beliebige Analyseprogrammiersprache integrieren und für datenbankinterne Analysen verwenden.
- Konnektivität: Exasol unterstützt Standardschnittstellen für die Integration von vor- und nachgelagerten (BI-)Tools. Zu den Standardschnittstellen, die mit der Exasol-Software ausgeliefert werden, gehören ODBC, JDBC, ADO.NET. Exasol lässt sich problemlos mit BI-Tools und Datenintegrations-Tools aller führenden Anbieter integrieren, wie z. B. Informatica, Talend, Pentaho, Tableau, Business Objects, Cognos, Power BI und Microstrategy. Zudem lässt es sich auch einfach mit gängigen SQL-Clients wie DBVisualizer, und DBeaver verbinden.
- Flexible Bereitstellungsoptionen: Die flexible Architektur von Exasol bietet eine Vielzahl von Bereitstellungsoptionen: als reine Softwarelösung, als Appliance oder in der Cloud (Exacloud, Microsoft Azure, Amazon Web Services und Google Cloud Platform).

Die Kernarchitektur von Exasol umfasst eine, wie in Abbildung 18 zu sehen ist, webbasierte grafische Benutzeroberfläche (EXAoperation), ein Cluster-Management-System (EXAClusterOS), das ein Linux-Derivat ist, und ein Storage-Management-Modul (EXAStorage).

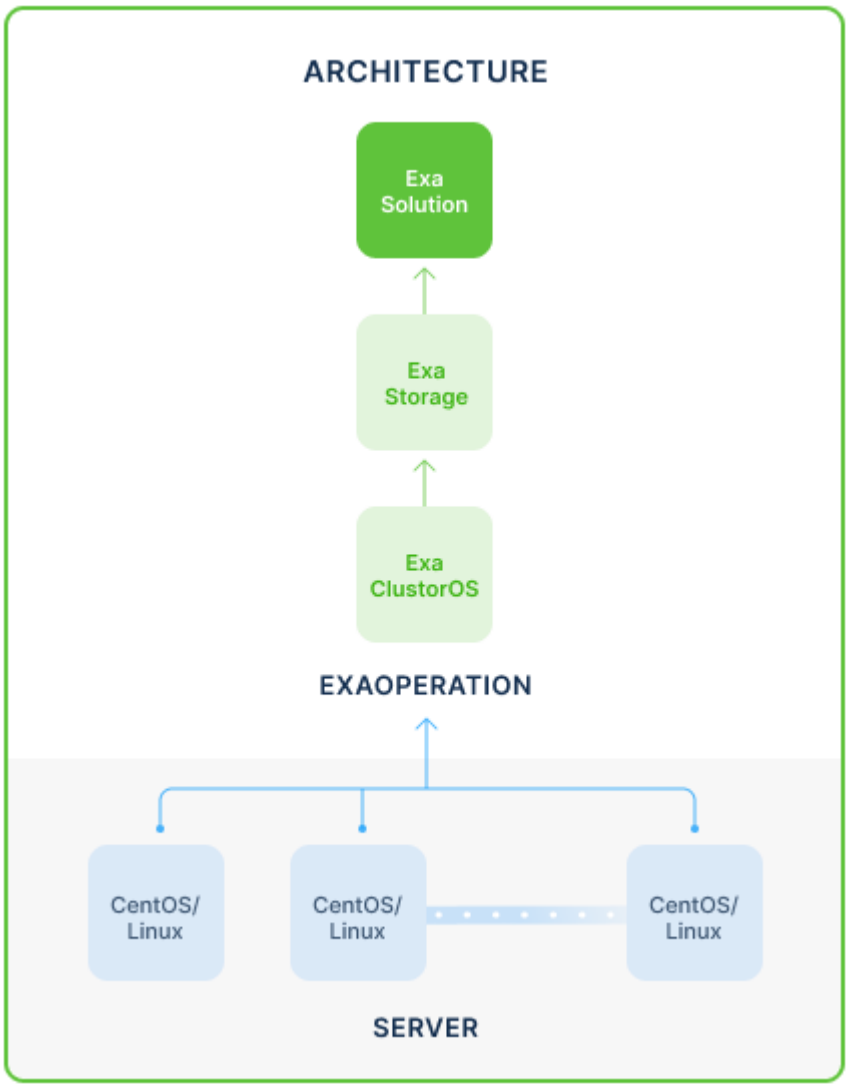


Abbildung 18: Exasol-Kernarchitektur

(Quelle: Exasol Overview, in: Exasol [Stichwort], o. J.)

Exasol unterstützt den ANSI-Standard SQL 2008 (einschließlich aller analytischen Funktionen) sowie eine große Auswahl des häufig verwendeten Oracle SQL-Dialekts. Exasol verfügt über einen hohen Automatisierungsgrad; dies deshalb, um sicherzustellen, dass es hohe Leistung bei minimalem Bedarf an teuren DBA(Datenbankadministrator)-Ressourcen für den Betrieb liefert. Wesentliche Schlüsselbereiche der Automatisierung sind:

- Automatische Verteilung von Daten: verteilt die Daten gleichmäßig auf alle Server im Cluster;
- automatische Duplizierung von Daten: dupliziert Daten automatisch über die Server hinweg, um die Datenintegrität im Falle eines Serverausfalls zu gewährleisten;
- automatische Auswahl der Kompressionsalgorithmen: Exasol wählt automatisch die Komprimierungsalgorithmen aus, die datentypspezifisch und für die In-Memory-

Verarbeitung optimiert sind. Diese Algorithmen arbeiten auch unabhängig auf jedem Knoten, um eine optimale Leistung zu gewährleisten;

- automatische Datenkomprimierung: Die Daten werden auf Spaltenebene komprimiert, wobei identische Images im Hauptspeicher und auf persistenten Medien (Festplatten) gespeichert werden, um die Leistung zu optimieren;
- automatische Überwachung und Protokollierung von Systemressourcen: Exasol überwacht Systemressourcen wie Arbeitsspeicher, Festplatte und CPU zur Unterstützung der Kapazitätsplanung.

3.5.3 Snowflake

Die Data Cloud von Snowflake basiert auf einer fortschrittlichen Datenplattform, die als SaaS bereitgestellt wird. Die Datenplattform von Snowflake baut nicht auf einer bestehenden Datenbanktechnologie oder Big Data-Softwareplattformen wie Hadoop auf. Stattdessen kombiniert Snowflake eine neue SQL-Abfrage-Engine mit einer innovativen Architektur, die nativ für die Cloud entwickelt wurde. Für den Benutzer bietet Snowflake, zusammen mit vielen zusätzlichen speziellen Funktionen, die gesamte Funktionalität einer analytischen Unternehmensdatenbank.

Bei Snowflake gibt es keine Hardware (weder virtuell noch physisch), die ausgewählt, installiert, konfiguriert oder verwaltet werden muss. Laufende Wartung, Verwaltung, Upgrades und Tuning werden von Snowflake übernommen. Snowflake läuft vollständig auf einer Cloud-Infrastruktur. Alle Komponenten des Snowflake-Services werden in öffentlichen Cloud-Infrastrukturen ausgeführt. Für den Rechenbedarf verwendet Snowflake virtuelle Recheninstanzen und einen Speicherdienst für die dauerhafte Speicherung von Daten. Snowflake kann nicht auf privaten Cloud-Infrastrukturen (weder vor Ort noch gehostet) ausgeführt werden und ist auch kein Software-Paket, das von einem Benutzer installiert werden kann, sondern Snowflake verwaltet selbst alle Aspekte der Softwareinstallation und -aktualisierung.

Die Architektur von Snowflake ist ein Hybrid aus traditionellen Shared Disk- und Shared Nothing-Datenbankarchitekturen. Ähnlich wie bei Shared Disk-Architekturen verwendet Snowflake ein zentrales Daten Repository für persistierte Daten, auf das von allen Rechenknoten der Plattform aus zugegriffen werden kann. Ähnlich wie bei Shared Nothing-Architekturen verarbeitet Snowflake jedoch Abfragen mithilfe von MPP-Rechenclustern, bei denen jeder Knoten im Cluster einen Teil des gesamten Datensatzes lokal speichert. Dieser Ansatz bietet die Einfachheit der Datenverwaltung einer Shared Disk-Architektur, jedoch mit

der Leistung und den Scale Out-Vorteilen einer Shared Nothing-Architektur. Abbildung 19 stellt die Kernarchitektur von Snowflake dar:

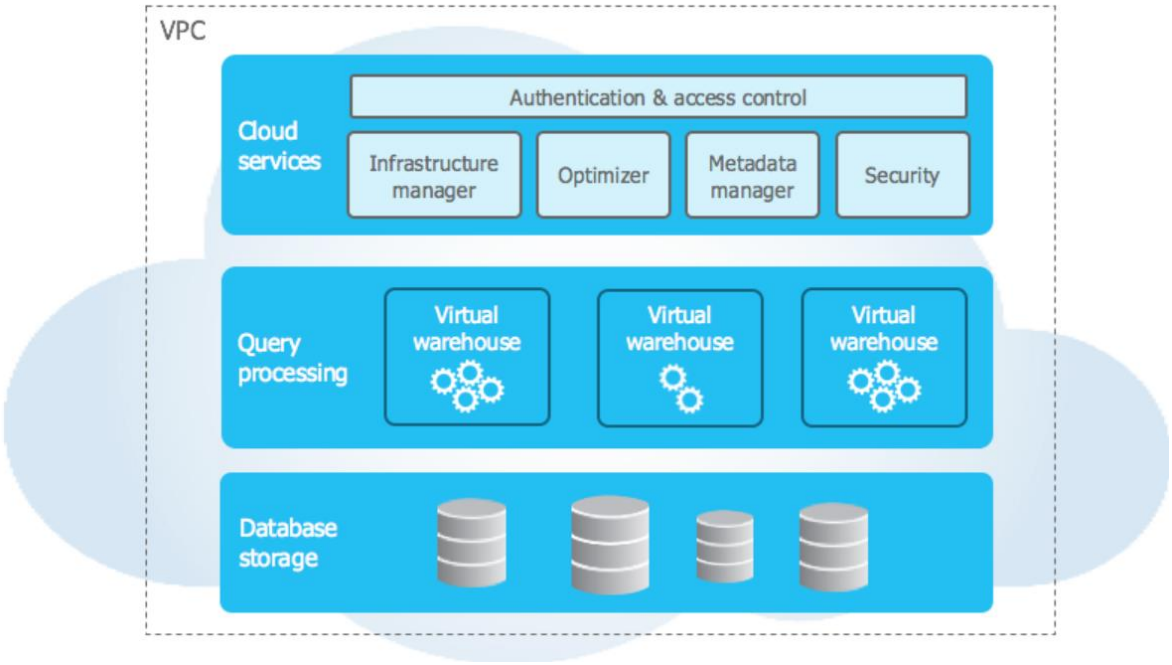


Abbildung 19: Snowflake-Kernarchitektur

(Quelle: Key Concepts & Architecture, in: Snowflake [Stichwort], o. J.)

Nachdem die Daten in Snowflake geladen wurden, reorganisiert Snowflake diese Daten in sein internes optimiertes, komprimiertes und spaltenförmiges Format. Snowflake speichert diese optimierten Daten in einem Cloud-Speicher. Dabei verwaltet Snowflake alle Aspekte, wie diese Daten gespeichert werden: die Organisation, Dateigröße, Struktur, Komprimierung, Metadaten, Statistiken und andere Aspekte der Datenspeicherung. Die von Snowflake gespeicherten Datenobjekte sind für Kunden weder direkt sichtbar noch zugänglich; sie sind nur über diejenigen SQL-Abfrageoperationen zugänglich, die mit Snowflake ausgeführt werden.

Abfragen werden in der Verarbeitungsschicht durchgeführt. Snowflake verarbeitet solche Abfragen mithilfe von „virtuellen Warehouses“. Jedes virtuelle Warehouse ist ein MPP-Rechencluster, der aus mehreren Rechenknoten besteht, die Snowflake von einem Cloud-Anbieter zugewiesen werden. Jedes virtuelle Warehouse ist ein unabhängiger Compute-Cluster, der sich keine Compute-Ressourcen mit anderen virtuellen Warehouses teilt. Folglich nimmt kein virtuelles Warehouse Einfluss auf die Leistung anderer virtueller Warehouses.

Die Schicht der Cloud-Dienste besteht aus einer Sammlung von Diensten, die Aktivitäten in Snowflake koordinieren. Diese Dienste verbinden alle verschiedenen Komponenten von Snowflake miteinander: von der Anmeldung bis zum Abfrageversand, um Benutzeranfragen

zu verarbeiten. Die Cloud-Serviceschicht läuft auch auf Recheninstanzen, die Snowflake vom Cloud-Anbieter bereitstellt. Zu den in dieser Schicht verwalteten Diensten gehören:

- Authentifizierung,
- Verwaltung der Infrastruktur,
- Verwaltung von Metadaten,
- Abfrageparsing und -optimierung
  und
- Zugriffskontrolle.

Snowflake wird als SaaS bereitgestellt, die vollständig auf einer Cloud-Infrastruktur läuft. Dies bedeutet, dass alle drei Schichten der Snowflake-Architektur – Speicher, Rechenleistung und Cloud-Dienste – vollständig auf einer ausgewählten Cloud-Plattform bereitgestellt und verwaltet werden. Ein Snowflake-Konto kann auf einer der nachfolgend~~en~~ genannten Cloud-Plattformen gehostet werden:

- Amazon Web Services (AWS)
- Google Cloud-Plattform (GCP)
- Microsoft Azure (Azure).

Auf jeder Plattform bietet Snowflake eine oder mehrere Regionen an, in denen das Konto bereitgestellt wird. Unterschiede bei den Kosten für Credits und den Datenspeicher werden auf jeder Cloud-Plattform nach der Region berechnet. Mittels einer oder mehrerer Region/en kann ausgewählt werden, wo Daten geografisch gespeichert werden, und zwar regional, national sowie international. Je nach gewählter Region wird festgelegt, wo Rechenressourcen bereitgestellt werden. Snowflake unterstützt mehrere Regionen für alle von Snowflake unterstützten Cloud-Plattformen, die in drei globale geografische Segmente (Nordamerika, Europa und Asien-Pazifik) unterteilt sind.

Snowflake bietet mehrere Editionen, die ausgewählt werden können, um sicherzustellen, dass die Nutzung den spezifischen Anforderungen entspricht. Jede nachfolgende Edition baut auf der vorherigen Edition auf, indem sie um editionsspezifische Funktionen und/oder höhere Service-Levels erweitert wird. Falls sich Anforderungen der Organisation ändern und wachsen, ist ein Wechsel der Editionen problemlos möglich. Snowflake bietet folgende Editionen an:

1. Standard-Edition

   Die Standard-Edition ist das Einstiegsangebot und bietet vollen, unbegrenzten Zugriff auf alle Standardfunktionen von Snowflake. Sie gewährleistet ein ausgewogenes Verhältnis zwischen Funktionen, Support-Level und Kosten.

2. Enterprise-Edition

   Die Enterprise-Edition bietet alle Funktionen und Services der Standard Edition, mit zusätzlichen Funktionen, die speziell für die Anforderungen von Großunternehmen und Organisationen entwickelt wurden.

3. Business-Critical-Edition

   Die Business-Critical-Edition, früher bekannt als Enterprise for Sensitive Data (ESD), bietet ein noch höheres Maß an Datenschutz, um die Anforderungen von Organisationen mit sensiblen Daten zu erfüllen, insbesondere PHI (Protected Health Information)-Daten, die den HIPAA- und HITRUST CSF-Vorschriften entsprechen müssen. Sie enthält alle Funktionen und Services der Enterprise Edition, zusätzlich aber noch mehr Sicherheit und Datenschutz. Darüber hinaus bietet sie mit Datenbank-Failover/Failback Unterstützung für Business Continuity und Disaster Recovery.

4. Virtuelle Private Snowflake (VPS)

   Virtual Private Snowflake bietet die höchste Sicherheitsstufe für Organisationen mit den strengsten Anforderungen, wie z. B. Finanzinstitute und alle anderen großen Unternehmen, die hochsensible Daten sammeln, analysieren und weitergeben. Sie umfasst alle Funktionen und Services der Business Critical Edition, jedoch in einer vollständig separaten Snowflake-Umgebung, die von allen anderen Snowflake-Konten isoliert ist (d. h. VPS-Konten teilen keine Ressourcen mit Konten außerhalb des VPS).

Snowflake stellt jede Woche eine neue Version bereit. So werden regelmäßig Serviceverbesserungen in Form von neuen Funktionen, Erweiterungen und Fehlerbehebungen bereitgestellt. Die Bereitstellung erfolgt im Hintergrund, so dass für die Benutzer keine Ausfallzeiten oder Unterbrechungen des Dienstes entstehen (vgl. Key Concepts & Architecture, in: Snowflake [Stichwort], o. J.; Supported Cloud Platforms, in: Snowflake [Stichwort], o. J.; Supported Cloud Regions, in: Snowflake [Stichwort], o. J.; Snowflake Editions, in: Snowflake [Stichwort], o. J.; sowie Snowflake Releases, in: Snowflake [Stichwort], o. J.).

4 Fallstudie

Nach der Darstellung der informationstechnischen Grundlagen sowie der verwendeten Technologien und Werkzeuge erfolgen im Anschluss daran die Beschreibung der Fallstudie, die Erläuterung und Begründung der verwendeten Methoden, die Darstellung des Benchmarks, die Nennung von Annahmen und Rahmenbedingungen sowie die Vorstellung der Vergleichsarchitektur. Darauf aufbauend wird zunächst ein Performancevergleich durchgeführt; in einem weiteren Schritt wird die avisierte Nutzwertanalyse vorgenommen.

4.1 Beschreibung

Im Rahmen des sich nun anschließenden Kapitels werden die Performance und der Nutzen eines Data Warehouse-Automatisierungstools unter Verwendung verschiedener Zielplattformen, konkret: Azure SQL-Datenbank, Exasol und Snowflake in der Azure Cloud untersucht. Dabei wird ein DWH mit dem TPC-H-Datensatz als Grundlage der Vergleichsarchitektur auf den verschiedenen Zielplattformen mit dem DVB implementiert. Die zu vergleichenden Implementierungen werden mit Cloud Computing Services von Microsoft umgesetzt. Die Betrachtung wird mit vier verschiedenen Performancemessungen und der Nutzwertanalyse erfolgen. Um der mit der DWH-Lösung einhergehenden Unsicherheit in der Analyse zu entsprechen, werden die Performancemessungen mehrfach wiederholt; zudem werden unterschiedlich große Datensätze verwendet.

4.2 Verwendete Methoden

Die Leistung beziehungsweise die Performance ist in der Fachliteratur definiert als eine Aussage über die Fähigkeit eines Anwendungssystems, eine Aufgabe oder eine Menge von Aufgaben innerhalb einer bestimmten Zeitspanne zu bewältigen (vgl. Heinrich, et al., 1998: 324). Sie ist damit ein entscheidendes Qualitätskriterium. Von schlechter Performance wird dann gesprochen, wenn ein Anwendungssystem die ursprünglichen Erwartungen oder Anforderungen der Anwender an die Performanceeigenschaften nicht erfüllt. Performance kann sowohl qualitativ als auch quantitativ beschrieben werden. Qualitative Aussagen, die z. B. durch Benutzerbefragungen gewonnen werden, sind meist subjektiv und nicht quantifiziert. Daher sind sie für eine wissenschaftliche Untersuchung zunächst unzureichend. Es bedarf einer möglichst genauen Quantifizierung der Performance. Quantitativ lässt sich die Performance mit Hilfe sogenannter Performancegrößen beschreiben, die sich durch Performancewerte des untersuchten Anwendungssystems operationalisieren lassen. Performancegrößen werden als

spezifische Produktmetriken aufgefasst, die auf verschiedenen Systemebenen und aus unterschiedlichen Perspektiven ermittelt werden können. So können Performancegrößen sowohl in Bezug auf einzelne Komponenten eines Anwendungssystems, z. B. eine CPU, als auch in Bezug auf eine komplette Systemumgebung entwickelt werden. Es können durchsatzbezogene (z. B. Transaktionsrate), auslastungsbezogene (z. B. CPU-Auslastung) und zeitbezogene (z. B. Antwortzeit) Performancegrößen definiert werden.

Eine andere Sichtweise zielt nur auf die Unterscheidung zwischen interner und externer Performance ab. Unter externer Performance wird diejenige Zeit verstanden, die ein Anwendungsprozess inklusive aller dafür notwendigen grundlegenden Systemprozesse beansprucht. Die interne Performance hingegen betrachtet einzelne IT-Ressourcen, wie z. B. eine CPU oder ein Netzwerk. Im Fall integrierter Informationssysteme ist eine solche Aussage jedoch wenig aussagekräftig, da die Prozesse datenbankbasierter Anwendungssysteme durch eine Vielzahl von Basisprozessen mit unterschiedlicher Ressourcenauslastung gekennzeichnet sind. Die interne Performance kann daher nicht als direktes Qualitätskriterium der Softwaretechnik angesehen werden. Sie wird aber häufig als Verrechnungsgröße im IT-Controlling der Systembetreiber verwendet (vgl. Rautenstrauch und Scholz, 1999: 262). Diese Charakterisierung der internen Performance legt es nahe, dass die nun folgende Performanceanalyse sich der externen Performance widmet. Untersuchungsgegenstand werden die Ingest Load-Performance, die Raw Vault Load-Performance und die durch den standardisierten TPC-H Benchmark vordefinierten 22 Abfragen durch eine Query-Performance sein.

Der Vergleich mittels qualitativer Aspekte soll wiederum die Erfahrungen, die bei der Implementierung der Zielplattformen gemacht werden, berücksichtigen. Hierzu gehört zum einen der Aufwand, der mit der Implementierung verbunden ist. Zum anderen werden auch der Administrationsaufwand, die Usability, die Flexibilität, die Skalierbarkeit und das erforderliche Fachwissen in diesen qualitativen Vergleich mit einbezogen. Lassen sich Aspekte nicht ausschließlich anhand von Zahlen vergleichen, besteht die Möglichkeit des Einsatzes sogenannter Scoringmodelle, wie beispielsweise die Nutzwertanalyse, die in der vorliegenden Arbeit Anwendung findet.

Im Folgenden werden nun diejenigen Methoden detailliert betrachtet, die im Kontext dieser Arbeit verwendet werden. Die Wahl des Benchmarking als Performancemessung und der Nutzwertanalyse als Scoringmodell lässt sich auf die breite Akzeptanz dieser Methoden zurückführen.

4.2.1 Benchmarking

Datenbank-Benchmarks sind ein wichtiges Werkzeug für Datenbankforscher und -praktiker. Sie erleichtern den Prozess, fundierte Vergleiche zwischen unterschiedlicher Datenbank-Hardware, -Software und -Konfigurationen vorzunehmen. Standard-Benchmarks sind dahingehend weit verbreitet, um die Leistung verschiedener Systeme zu vergleichen und die allgemeine Frage zu beantworten: „Welches ist das beste System in einer bestimmten Domäne für bestimmte Arten von Anwendungen?“ Benchmarking ist eine Schlüsselaktivität beim Aufbau und Tuning von Datenverwaltungssystemen. Daher ist es die Aufgabe eines Benchmarks, zu erfassen, wie gut verschiedene Systeme im Hinblick auf die Zielsetzung unter den gegebenen Rahmenbedingungen abschneiden. In der Praxis werden Benchmarks dazu eingesetzt, um Entscheidungen über die wirtschaftlichste Bereitstellungsstrategie zu unterstützen und um Erkenntnisse über Leistungsengpässe zu gewinnen.

Einige der am häufigsten verwendeten Datenbank-Benchmarks stammen vom Transaction Processing Performance Council (TPC), einer Non-Profit-Organisation, deren Hauptziel es ist, Kriterien für die Gewinnung von Informationen über die Leistung von Datenbanken durch die Verwendung von Benchmarks festzulegen. Das TPC ist derzeit die maßgebliche Instanz für die Erstellung von Benchmarks, die in Bereichen wie Transaktionsverarbeitung und Datenbankanwendungen eingesetzt werden. Das TPC wurde 1988 gegründet, um Computerherstellern zu ermöglichen, ihre Produkte anhand eines Standardsatzes von Tests, den sogenannten Benchmarks, zu vergleichen. An ihm beteiligt sind 16 Unternehmen. Das TPC ist auch an der Schaffung von Prozessen zur Überwachung und Überprüfung der Gültigkeit von bestehenden Benchmarks beteiligt, um zu verhindern, dass Unternehmen ihre Ergebnisse zu sehr „aufblasen“. Die vom TPC-Konsortium definierten Benchmarks und ihre Derivate stellen den De-facto-Standard bei der Bewertung und dem Vergleich der Leistung von Datenbankmanagementsystemen dar (für den gesamten vorangehenden Absatz vgl. Barata, et al., 2015: 619 f.).

4.2.2 Nutzwertanalyse

Die Nutzwertanalyse gehört zu den sogenannten mehrdimensionalen Verfahren der Nutzenbewertung. Ihr Begründer, Christof Zangemeister, definiert die Nutzwertanalyse als „die Analyse einer Menge komplexer Handlungsalternativen mit dem Zweck, die Elemente dieser Menge entsprechend den Präferenzen des Entscheidungsträgers bezüglich eines

multidimensionalen Zielsystem zu ordnen“ (Zangemeister, 2014: 45[3]). Diese Anordnung erfolgt durch die Berechnung der Nutzenwerte vorhandener Alternativen.

Zunächst werden gewichtete Zielwerte entsprechend ihrer Relevanz für den Entscheider berücksichtigt. Die Erreichung der einzelnen Ziele wird an den verschiedenen Alternativen gemessen und in Form eines Teilnutzenwertes angegeben. Die Eigenschaften der Entscheidungsalternativen sollen möglichst objektiv und eindeutig beschrieben werden können. Unter Berücksichtigung der Kriteriengewichtung werden die Teilnutzenwerte zu einem Gesamtwert für jede Alternative, dem sogenannten Nutzwert, zusammengefasst.

Durch die Summierung der Teilnutzenwerte können ungünstige Ausprägungen hinsichtlich einzelner Zielwerte ausgeglichen werden; dabei ist zu prüfen, ob für bestimmte Kriterien Mindestanforderungen bestehen oder ob es K.-o.-Kriterien gibt (vgl. Götze, 2014: 193). K.-o.-Kriterien stellen absolute Mindestanforderungen dar, die von allen Alternativen in vollem Umfang erfüllt werden müssen. Wenn eine Alternative auch nur ein K.-o.-Kriterium nicht erfüllt, muss sie als Alternative ausgeschlossen werden (vgl. Gruschka und Schnitzenbaumer, 2008: 84).

Nach (Blohm, et al., 2013: 151 f.) und (Lücke, 1991: 296) sind die Verfahrensschritte einer Nutzwertanalyse wie folgt:

1. Bestimmung der Zielkriterien
2. Gewichtung der Zielkriterien
3. Teilnutzenbestimmung
4. Bestimmung des Nutzwerts
   und
5. Bewertung des Nutzwerts.

Bei der Festlegung der Zielkriterien ist darauf zu achten, dass die Zielkriterien operational formuliert sind, d. h., dass für jedes Kriterium eine Mess-Skala erforderlich ist, an der die Zielerreichung mit einem nominalen, ordinalen oder kardinalen Skalenniveau gemessen werden kann. Die Mehrfacherfassung von Projektmerkmalen ist zu vermeiden und in diesem Zusammenhang die Nutzenunabhängigkeit aller Zielkriterien sicherzustellen. Nutzenunabhängigkeit ist definiert als die Erreichung eines Zielkriteriums, ohne dass die Erfüllung eines anderen Zielkriteriums erforderlich ist (vgl. Blohm, et al., 2013: 152). Um die Zielkriterien zu

[3] Mir ist bewusst, dass Literatur aus einem Selbstverlag üblicherweise als nicht zitierfähig gilt. Aufgrund der Tatsache, dass dieser Autor als Begründer der Nutzwertanalyse gilt, und nach genauer Prüfung der Fakten, habe ich mich entschlossen, diesen Titel trotz des gegebenen Vorbehalts als Quelle zu verwenden.

bestimmen, sind in der Regel eine umfangreiche und sorgfältige Analyse und Strukturierung des Zielsystems erforderlich.

Durch die Gewichtung der Zielkriterien bekommen die Kriterien eine unterschiedlich hohe Bedeutung. Für die Gewichtung können verschiedene Skalierungsmethoden verwendet werden, z. B. die direkte oder indirekte Intervallskalierung (vgl. Blohm, et al., 2013: 154 f.; und Zangemeister, 2014: 163-175).

Bei der direkten Intervallskalierung werden den Zielkriterien Werte aus einer Intervallskala zugewiesen, deren Intervalle die Präferenzunterschiede der Entscheidungsträger widerspiegeln. Bei der indirekten Intervallskalierung wird zunächst eine Rangreihe der Zielkriterien gebildet. Entsprechend der Position in der Rangreihe wird jedem Zielkriterium eine Rangnummer zugeordnet, wobei der Anwender dem wichtigsten Kriterium den Wert K (K = Anzahl der Kriterien) und dem unwichtigsten Kriterium den Wert 1 zuordnet. Der zweite Schritt besteht darin, die Rangreihe in eine Intervallskala zu transformieren, wobei eine Annahme von Präferenzunterschieden zwischen aufeinanderfolgenden Rangreihen notwendig ist. Wird nach der obigen Zuordnung von gleichen Präferenzdistanzen zwischen jeweils zwei aufeinanderfolgenden Kriterien in der Rangreihe ausgegangen, können die Rangzahlen als Gewichte verwendet werden (vgl. Zangemeister, 2014: 172 f.).

Der Teilwert wird für jede Alternative und jedes Kriterium der untersten Hierarchiestufe ermittelt. Um eine größtmögliche Nachvollziehbarkeit der Nutzwertanalyse zu gewährleisten, sollte die Ermittlung von Teilnutzen möglichst nicht in einem einzigen Schritt durch subjektive Bewertung und ohne explizite Angabe der Erreichung der einzelnen Zielkriterien erfolgen. Geeigneter (wenn auch aufwendiger) geschieht dies in zwei Schritten (vgl. Blohm, et al., 2013: 158): Im ersten Schritt werden die Werte der einzelnen Alternativen bezüglich der Zielkriterien mit Hilfe von Nominal-, Ordinal- oder Kardinalskalen ermittelt; im zweiten Schritt werden die gemessenen Zielerreichungswerte jeweils in einen Teilnutzenwert transformiert, der mit einer Teilnutzenskala gemessen wird. Da die Transformation der Zielerreichungswerte in Teilnutzenwerte einer subjektiven Beurteilung unterliegt, sollte sie aus Gründen der Nachvollziehbarkeit mit Hilfe von Transformationsfunktionen durchgeführt werden. Diese lassen sich in drei verschiedene Typen einteilen (vgl. Blohm, et al., 2013: 159-161):

1. Bei der diskreten Transformationsfunktion wird dem Zielerreichungswert genau ein einziger Teilnutzenwert zugeordnet: Eine ordinale Skalierung der Zielleistung reicht aus, um die Transformationsfunktion zu formulieren. 2. Bei der stückweise-konstanten Transformationsfunktion bleibt der Teilnutzenwert für feste Intervalle der Zielerreichungswerte konstant. Um

eine solche Transformationsfunktion zu formulieren, ist eine kardinale Skalierung erforderlich. 3. Der dritte Typ von Transformationsfunktionen, die stetigen Transformationsfunktionen, unterscheidet sich von stückweise-konstanten Transformationsfunktionen dadurch, dass die einzelnen Intervalle gegen Null gehen und die Anzahl der Intervalle gegen Unendlich geht und dass schon kleine Unterschiede in den Zielerreichungswerten jeweils zu unterschiedlichen Teilnutzenwerten führen. Dies erfordert wiederum eine kardinale Skalierung. Durch diesen Schritt der Nutzwertanalyse wird die Annahme impliziert, dass nämlich die Zielerreichung korrekt gemessen und entsprechend der Präferenz der Entscheidungsträger in einen Teilnutzwert umgerechnet wird (vgl. Götze, 2014: 195 f.).

Wenn, wie im Folgenden angenommen, die Zielkriterien nutzwertunabhängig oder bedingt nutzwertunabhängig sind und eine einheitliche kardinale Skala zur Messung der Teilwerte verwendet wird, sieht die additive Funktion, wie nachfolgend dargestellt, aus:

$$N_{Ni} = \sum_{k=1}^{K} n_{ik} \cdot w_k$$

*Formel *: Additive Funktion*

(Quelle: Götze, 2014: 196)

Die Teilnutzenwerte n_{ik} der Alternativen i zu den Kriterien k der untersten Hierarchiestufe werden durch Multiplikation mit den Kriteriengewichten w_k vergleichbar gemacht und anschließend durch Addition zusammengefasst. Der Nutzwert N_{Ni} einer Alternative i ist die Summe der gewichteten Teilnutzenwerte (vgl. Götze, 2014: 196).

Um die Vorteilhaftigkeit von Untersuchungsobjekten zu bewerten, gelten folgende Vorteilhaftigkeitsregeln:

1. Absolut vorteilhaft ist ein Untersuchungsobjekt dann, wenn ihr Nutzwert einen festgelegten Schwellenwert überschreitet.

2. Ein Untersuchungsobjekt ist dann relativ vorteilhaft, wenn sein Nutzen größer ist als derjenige aller anderen zur Auswahl stehenden Untersuchungsobjekte.

4.3 Der Benchmark

Der TPC-Benchmark (TPC-H) ist ein Benchmark zur Entscheidungsunterstützung. Er besteht aus einer Reihe von geschäftsorientierten Ad-hoc-Abfragen und gleichzeitigen Datenänderungen. Die Abfragen und die Daten, welche die Datenbank befüllen, wurden so ausgewählt, dass sie eine breite branchenweite Relevanz haben und gleichzeitig ein ausreichendes Maß an

einfacher Implementierung aufweisen. Der Benchmark bildet Entscheidungsunterstützungssysteme ab, die

- große Datenmengen untersuchen,
- Abfragen mit einem hohen Grad an Komplexität ausführen
  und
- Antworten auf kritische Geschäftsfragen geben.

TPC-H bewertet die Leistung verschiedener Entscheidungsunterstützungssysteme durch die Ausführung von Abfragesätzen gegen eine Standarddatenbank unter kontrollierten Bedingungen. Die TPC-H-Abfragen:

- geben Antworten auf reale Geschäftsfragen;
- simulieren generierte Ad-hoc-Abfragen (z. B. über eine Point-and-click-GUI-Schnittstelle);
- sind weitaus komplexer als die meisten OLTP-Transaktionen;
- umfassen eine große Bandbreite an Operatoren und Selektivitätseinschränkungen;
- erzeugen eine intensive Aktivität auf Seiten der Datenbankserver-Komponente des zu testenden Systems;
- werden gegen eine Datenbank ausgeführt, die bestimmte Populations- und Skalierungsanforderungen erfüllt

  und:

- werden mit solchen Einschränkungen implementiert, die sich aus der engen Synchronisation mit einer Online-Produktionsdatenbank ergeben.

Die Geschäfts- und Anwendungsumgebung des Benchmark besteht aus einer Reihe von Geschäftsabfragen, die so konzipiert sind, dass sie die Systemfunktionalitäten in einer Weise ausüben, die für komplexe Geschäftsanalyseanwendungen repräsentativ sind. Die Abfragen wurden in einen realistischen Kontext gestellt, der die Tätigkeit eines Großhandelslieferanten darstellt, um dem Anwender einen intuitiven Bezug zu den Komponenten des Benchmark zu ermöglichen. Der TPC-H-Benchmark repräsentiert keine bestimmte Geschäftsaktivität, sondern eine beliebige Branche, die den Verkauf oder die Verteilung eines Produkts weltweit verwalten muss (z. B. Autovermietung, Lebensmittelvertrieb, Teile, Zulieferer usw.).

Das Ziel im Zusammenhang mit dem Benchmark besteht darin, die Vielfalt der Operationen, die in einer Informationsanalyseanwendung vorkommen, zu reduzieren, wohingegen die wesentlichen Leistungsmerkmale der Anwendung erhalten bleiben, nämlich: der Grad der

Systemauslastung und die Komplexität der Operationen. Um eine Geschäftsanalyseumgebung vollständig zu kontrollieren, muss eine große Anzahl von Abfragen unterschiedlicher Komplexität ausgeführt werden. Viele der Abfragen sind nicht performanceoptimiert, da sie sehr lange laufen und viele Systemressourcen benötigen. Die Abfragen, die ausgewählt wurden, weisen die folgenden Eigenschaften auf:

- Sie haben einen hohen Grad an Komplexität;
- sie verwenden eine Vielzahl von Zugriffen;
- sie sind von Ad-hoc-Natur;
- sie untersuchen einen großen Prozentsatz der verfügbaren Daten
  und
- sie unterscheiden sich alle voneinander.

Diese ausgewählten Abfragen liefern Antworten auf die folgenden Kategorien von Geschäftsanalysen:

- Preisgestaltung und Promotionen;
- Angebots- und Nachfragesteuerung;
- Gewinn- und Umsatzmanagement;
- Studie zur Kundenzufriedenheit;
- Studie zum Marktanteil

  und

- Versandmanagement.

Andere TPC-Benchmarks modellieren das operative Ende der Geschäftsumgebung, in der Transaktionen auf Echtzeitbasis ausgeführt werden. Der TPC-H-Benchmark hingegen modelliert das Analyse-Ende der Geschäftsumgebung, in der Trends berechnet und verfeinerte Daten erzeugt werden, um fundierte Geschäftsentscheidungen treffen zu können. Abbildung 20 stellt das Geschäftsumfeld des TPC-H-Benchmark dar:

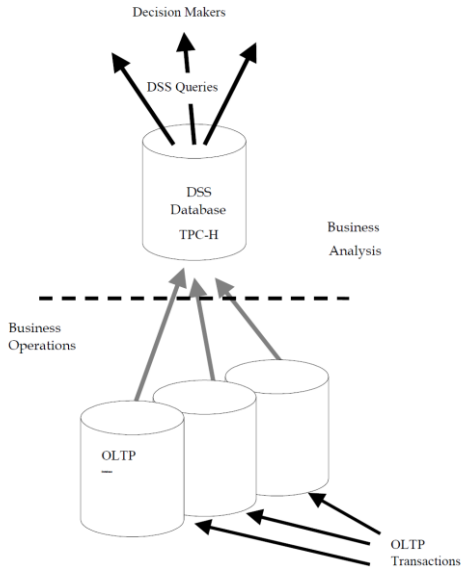


Abbildung 20: Geschäftsumfeld des TPC-H-Benchmark

(Quelle: Transaction Processing Performance Council [TPC], 2018: 12)

Die Komponenten der TPC-H-Datenbank sind so definiert, dass sie aus acht separaten und individuellen Tabellen bestehen (den sog. Basistabellen). Die Beziehungen zwischen den Spalten dieser Tabellen sind in Abbildung 21 dargestellt:

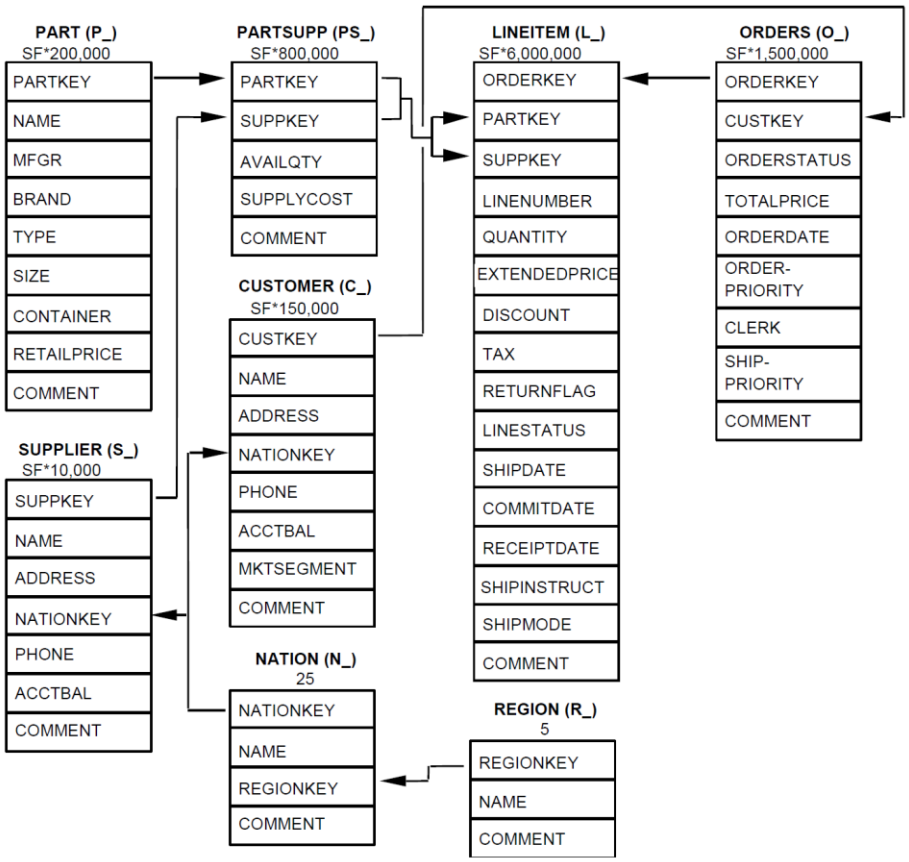


Abbildung 21: Das TPC-H-Schema

(Quelle: Transaction Processing Performance Council [TPC], 2018: 13)

Die Pfeile zeigen in die Richtung der One-to-many-Beziehungen zwischen den Tabellen. Die Zahl/Formel unter jedem Tabellennamen gibt die Kardinalität (Anzahl der Zeilen) der Tabelle an. Um die gewählte Datenbankgröße zu erhalten, sind einige Tabellen mit SF, dem Skalierungsfaktor, faktorisiert (vgl. Transaction Processing Performance Council [TPC], 2018: 8-13).

4.4 Rahmenbedingungen

Für jede Zieldatenbank läuft eine DVB-Instanz in der Version 5.4.6.0 auf einer Azure Virtual Machine mit einer öffentlichen IP-Adresse, um auf die GUI des DVB zuzugreifen. Als Betriebssystem der Virtual Machines wird Ubuntu 18.04.05 LTS (Long Term Support) verwendet. Die Server verfügen jeweils über zwei virtuelle CPU-Kerne und 32 GB RAM. Als Festplattenspeicher (Solid State Drive [SSD]) sind 32 GB verfügbar.

Die Zieldatenbanken werden in vier unterschiedlich großen Skalierungsstufen untersucht. Exasol wird als Single Node und nicht im Cluster betrieben und läuft auf der Datenbankversion 7.0.4., die Exasol als Image in Azure bereitstellt. Die Auswahl der verschiedenen Skalierungsgrößen hat sich dabei an der Auswahl von Snowflake orientiert, da Snowflake die restriktivsten Skalierungsoptionen bietet. Die verwendeten Skalierungsstufen der Zieldatenbanken sind der folgenden Tabelle 1 zu entnehmen:

Zielplattform	Skalierungsstufe[4]
Azure SQL-Datenbank	Allgemein: Serverlos, Gen5, 8 virtuelle Kerne
	Allgemein: Serverlos, Gen5, 16 virtuelle Kerne
	Allgemein: Serverlos, Gen5, 32 virtuelle Kerne
	Allgemein: Serverlos, Gen5, 40 virtuelle Kerne
Exasol	Standard D8s v3 (8 vcpus[5], 32 GiB Arbeitsspeicher)
	Standard D16s v3 (16 vcpus, 64 GiB Arbeitsspeicher)
	Standard D32s v3 (32 vcpus, 128 GiB Arbeitsspeicher)
	Standard D64s v3 (64 vcpus, 256 GiB Arbeitsspeicher)
Snowflake	XS = 1 VM --> 8 virtuelle Kerne
	S = 2 VM --> 16 virtuelle Kerne
	M = 4 VM --> 32 virtuelle Kerne
	L = 8 VM --> 64 virtuelle Kerne

Tabelle 1: Skalierungsstufen der Zieldatenbanken

(Quelle: Eigene Darstellung)

Für den Vergleich wird der TPC-H Datensatz in drei unterschiedlich großen Skalierungen verwendet: zum einen in der 1 GB-Variante und zum anderen in der 5 und 10 GB-Variante. Genauere Informationen zum TPC-H Datensatz wurden in Abschnitt 4.5 gegeben.

Die Performancemessung der Zieldatenbanken erfolgt mit vier unterschiedlichen Messungen:

1. Ingest Load-Performance: Die Laufzeit, die benötigt wird, um die Daten aus den Quellen in die jeweilige Zieldatenbank zu laden;
2. Raw Vault Load-Performance: Die Laufzeit, die benötigt wird, um die Daten aus dem Staging in das Raw Vault Model zu laden;
3. Query Performance (ohne Cache): Die Laufzeit, die für die 22 vordefinierten TPC-H-Abfragen ohne Cache benötigt wird;
4. Query Performance (mit Cache): Die Laufzeit, die für die 22 vordefinierten TPC-H-Abfragen mit Cache benötigt wird.

[4] Die Bezeichnung der Skalierungsstufen für die Azure SQL-Datenbank und Exasol wurde 1:1 der Dokumentation zu Microsoft Azure entnommen. Für Snowflake wurde die Anzahl der VM und der virtuellen Kerne ergänzend hinzugefügt.

[5] vCPU ist die Abkürzung für Virtual Centralized Processing Unit. Eine vCPU stellt einen Teil oder einen Anteil der zugrunde liegenden physischen CPU dar, der einer virtuellen Maschine (VM) zugewiesen ist.

Um valide Aussagen treffen zu können, wurden die Performancetests mehrfach durchlaufen und der Mittelwert angenommen. Für die 1 GB-Variante des TPC-H-Datensatzes wurden alle Messungen 10-mal wiederholt. Für die 5 und 10 GB-Variante des TPC-H-Datensatzes wurden lediglich zwei Skalierungsstufen der Zieldatenbanken (1 und 3) untersucht. Die Ingest Load-Performancemessung wurde zweimal wiederholt, die übrigen Messungen viermal, um die Kosten im Rahmen zu halten. Die Query-Performancemessungen wurden mit dem DBeaver als DBMS gemessen.

4.5 Vergleichsarchitektur

In diesem Abschnitt wird diejenige Vergleichsarchitektur vorgestellt, die die einzelnen Komponenten zeigt und eine objektive Vergleichbarkeit ermöglichen soll. Der Ansatz wird sich auf eine statische Sichtweise beschränken. Die Referenz- beziehungsweise Vergleichsarchitektur wird als Grundlage für die Performancemessungen der zu vergleichenden Implementierungen verwendet und dient den im Folgenden genannten Zwecken:

- Vergleich von Werkzeugen für Data Warehousing;
- Aufzeigen von Stärken und Schwächen des DWH-Systems;
- in der Lage zu sein, auf Basis der Vergleichsarchitektur eine konkrete Implementierung für ein DWH-System zu planen;
- einen Überblick über die einzelnen Komponenten zu geben

  und:

- zu einem gemeinsamen und klaren Verständnis zwischen Autor und Leser führen (vgl. Bauer und Günzel, 2013: 37-39).

Die Vergleichsarchitektur ist in Abbildung 22 dargestellt, welche die vorhandenen Operanden, visualisiert als Datenbehälter (Datenquellen und Arbeitsbereich) und Datenflüsse, die die Komponenten miteinander verbinden, zeigt. Die ‚Semantic'- und die ‚Reporting'-Sicht werden in der folgenden Architektur für die Performanceanalyse nicht berücksichtigt.

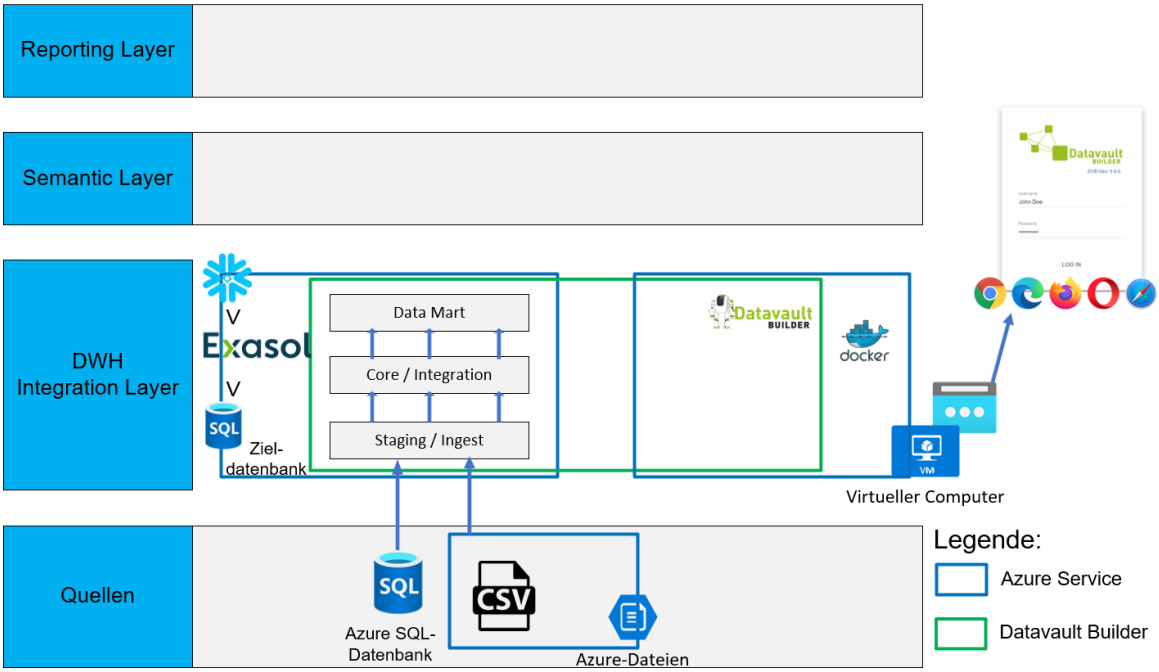


Abbildung 22: Vergleichsarchitektur

(Quelle: Eigene Darstellung unter Verwendung von [2150 GmbH - Features Overview, in: Datavault-builder {Stichwort}, o. J.; 2150 GmbH - Home, in: datavault-builder {Stichwort}, o. J.; CSV Logo, in: Flaticon {Stichwort}, o. J.; Docker Logo, in: Testautomatisierung {Stichwort}, o. J.; Exasol Vector Logo, in: Seekvectorlogo {Stichwort}, 2021; Irish, 2010; sowie Snowflake Computing Logo Clipart, in: Pikpng {Stichwort}, o. J.])

Die Quellen werden zum einem für den 1 GB TPC-H-Datensatz aus CSV-Dateien und zum anderen aus einer Azure SQL-Datenbank für den 5 und 10 GB TPC-H-Datensatz verarbeitet. Der DVB läuft auf mehreren Docker-Containern in einer Azure Virtual Machine. Auf die GUI des DVB wird über eine öffentliche IP-Adresse mit einem beliebigen Browser zugegriffen. Durch die Konnektoren des DVB kommen die Daten in die Ingest-Schicht und werden in der gewünschten Zielplattform abgelegt. Der DVB erstellt eine Data Vault-Architektur innerhalb der Zieldatenbank inklusive der notwendigen ETL/ELT-Prozesse. Das TPC-H-Datavault-Modell, welches für die Raw Vault Load-Performance verwendet wird, ist in Abbildung 23 dargestellt. Das modellierte logische Design wird anschließend in eine funktionierende, technische Implementierung umgesetzt.

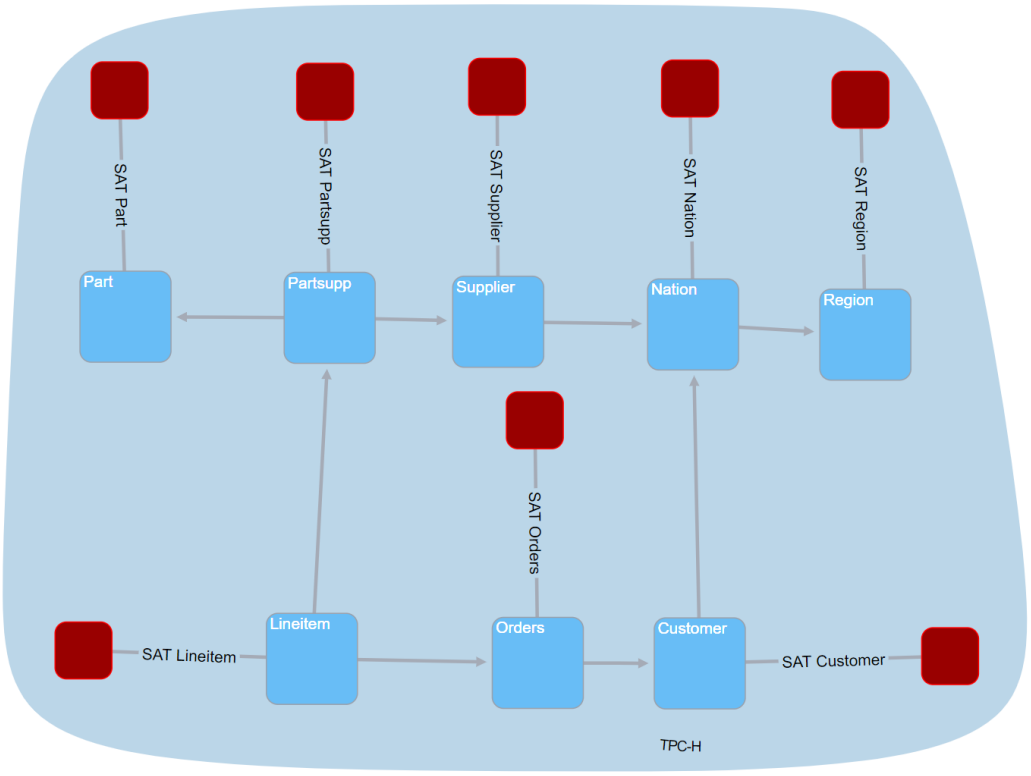


Abbildung 23: TPC-H-Datavault-Modell

(Quelle: Eigene Darstellung)

Die in der Abbildung dargestellten blauen Vierecke repräsentieren die Hubs, die roten Vierecke die Satelliten, und die Pfeile zwischen den Hubs stellen die Links dar.

5 Performance-Nutzen-Analyse

In dem sich nun anschließenden Kapitel wird eine Darstellung von Performance, Nutzen und den daraus resultierenden Ergebnissen der Bereitstellung der Vergleichsarchitektur auf verschiedene Zielplattformen in Kombination mit dem Datavault Builder vorgenommen.

5.1 Performanceanalyse

Die Ergebnisse der Performanceanalyse der Zieldatenbanken Azure SQL-Datenbank, Exasol und Snowflake werden jeweils pro unterschiedlicher Performancemessung und Datensatzgröße des TPC-H-Benchmark dargestellt. Die Diagramme zeigen die Gesamtlaufzeit (orange) der jeweiligen Performancemessung sowie die dadurch entstandenen Kosten (blau) auf. Die Beträge ergeben sich aus den Datenbankkosten pro Sekunde multipliziert mit der Gesamtlaufzeit. Die genaue Berechnung der Kosten ist Anhang 4 zu entnehmen. Es sei

anzumerken, dass es sich in den nachfolgenden Diagrammen um Bruttopreise handelt, deren Zahlen gewissenhaft gerundet wurden.

5.1.1 Ingest Load-Performance

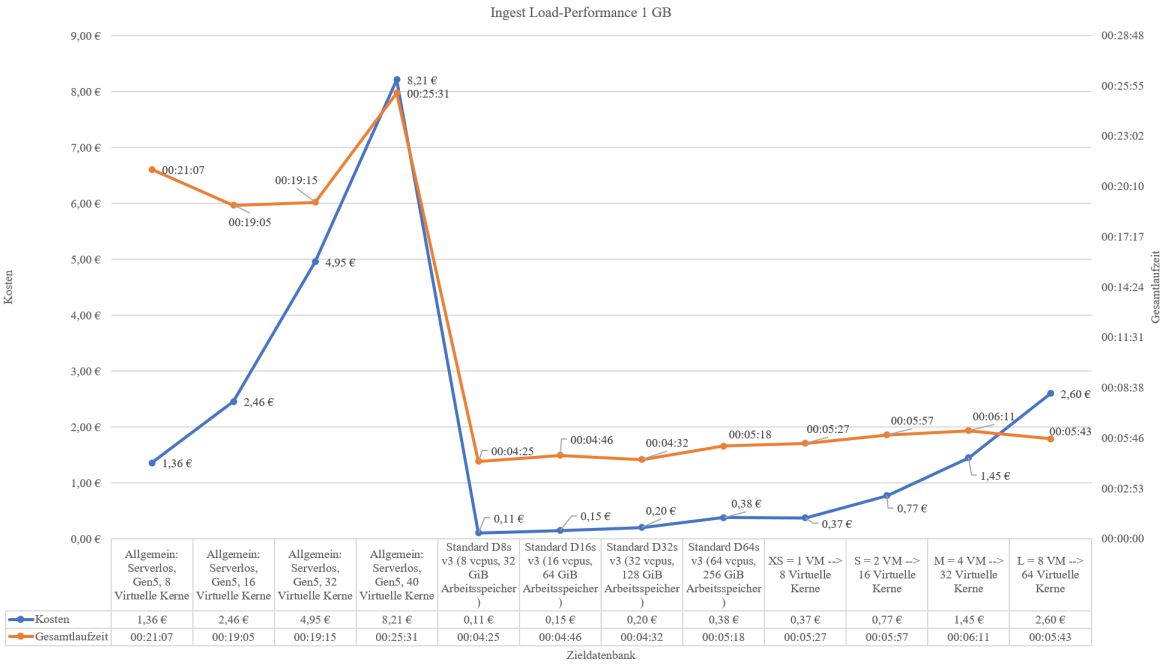


Abbildung 24: Ingest Load-Performance 1 GB[6]

(Quelle: Eigene Darstellung)

Aus der obenstehenden Abbildung 24 wird ersichtlich, dass eine höhere Skalierung der Datenbanken so gut wie keinen Einfluss auf die Gesamtlaufzeit hat. Abgesehen von der Azure SQL-Datenbank treten bei höheren Skalierungsstufen sogar höhere Gesamtlaufzeiten auf. Die Azure SQL-Datenbank ist deutlich langsamer als Exasol und Snowflake. Exasol ist im Vergleich zur Azure SQL-Datenbank und Snowflake am günstigsten und verursacht in der stärksten Skalierung in etwa so viele Kosten wie Snowflake in der kleinsten Skalierungsstufe.

[6] Hinweis: Zur Gewährleistung einer guten Lesbarkeit werden solche Abbildungen, deren inliegende Schrift im Original kaum oder nur schwer zu entziffern ist, dieser Projektarbeit als vergrößerter Anhang beigefügt.

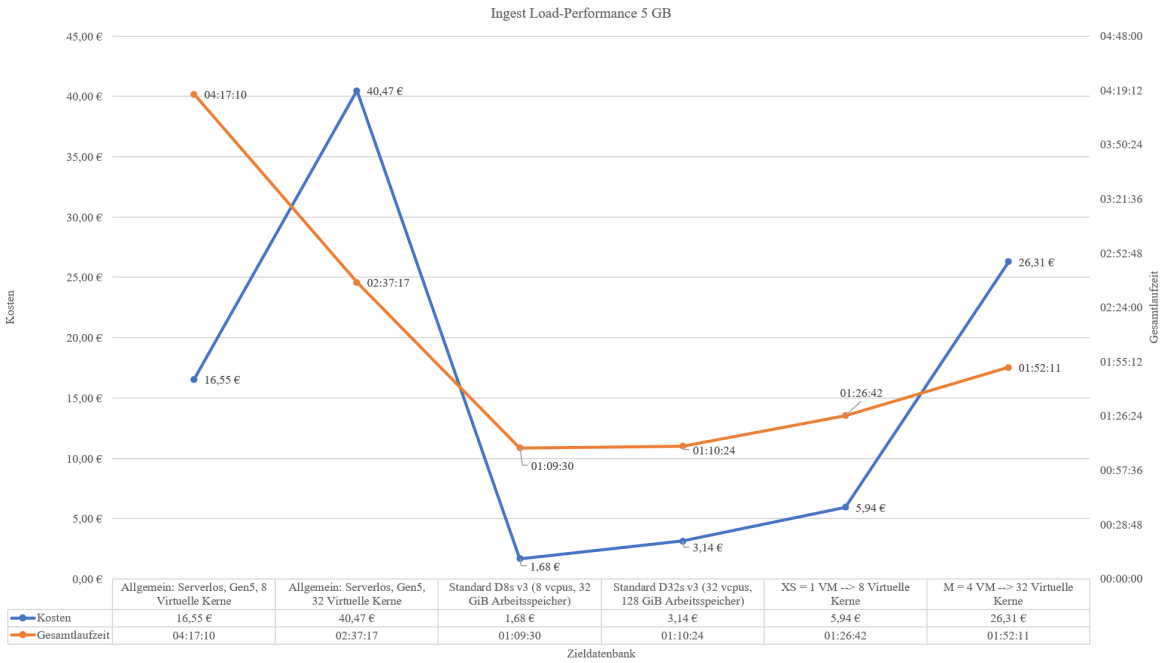


Abbildung 25: Ingest Load-Performance 5 GB

(Quelle: Eigene Darstellung)

Abbildung 25 zeigt, dass durch die Skalierung der Datenbanken die Kosten der Azure SQL-Datenbank um das ca. 2,5-Fache ansteigen und die Gesamtlaufzeit sich dadurch um das 1,7-Fache verringert. Für Exasol und Snowflake hat die Skalierung die Gesamtlaufzeit nicht verringert, sondern im Gegenteil verlängert. Im Vergleich zum 1 GB TPC-H-Datensatz ist die Azure SQL-Datenbank wieder am langsamsten und verursacht die höchsten Kosten. Exasol ist wieder am günstigsten.

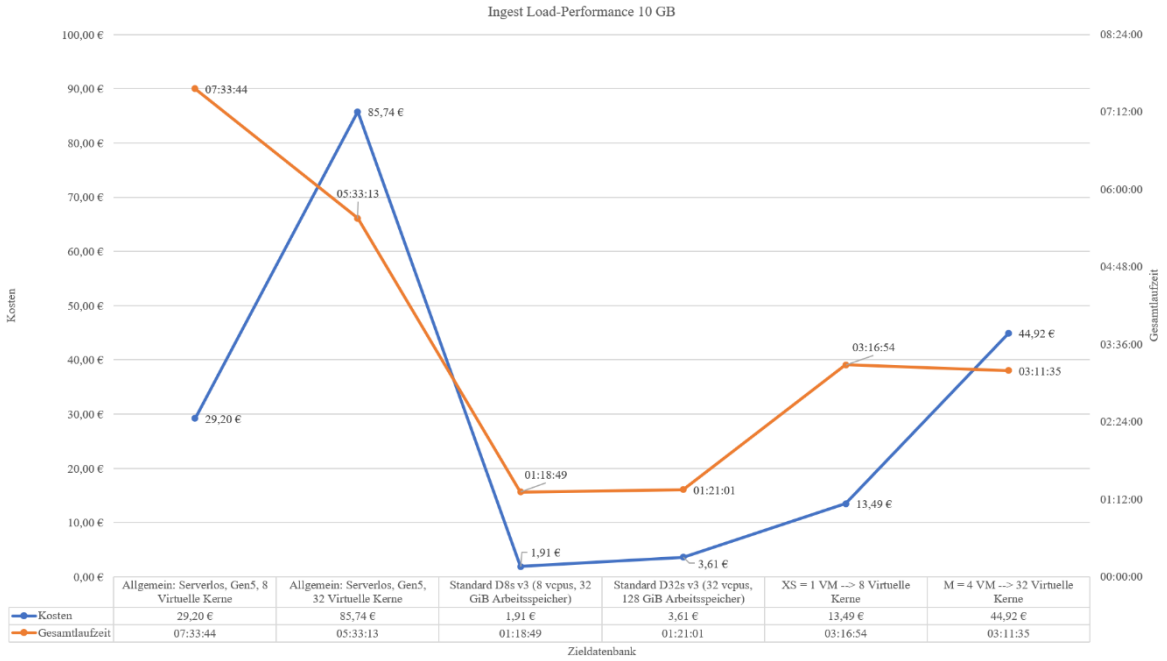


Abbildung 26: Ingest Load-Performance 10 GB

(Quelle: Eigene Darstellung)

Aus Abbildung 26 wird ersichtlich, dass durch die Skalierung der Datenbanken die Kosten der Azure SQL-Datenbank um das ca. 3-Fache ansteigen und die Gesamtlaufzeit sich dadurch um das 1,3-Fache verringert. Für Exasol bewirkt die Skalierung erneut keine Verringerung der Gesamtlaufzeit, sondern sie verlängert sie wieder. Snowflake ist durch die höhere Skalierung ca. 5 Minuten schneller, allerdings zu 3,3-fach höheren Kosten. Im Vergleich zum 1 und 5 GB TPC-H-Datensatz ist die Azure SQL-Datenbank wieder am langsamsten und verursacht die höchsten Kosten. Exasol ist deutlich am günstigsten.

5.1.2 Raw Vault Load-Performance

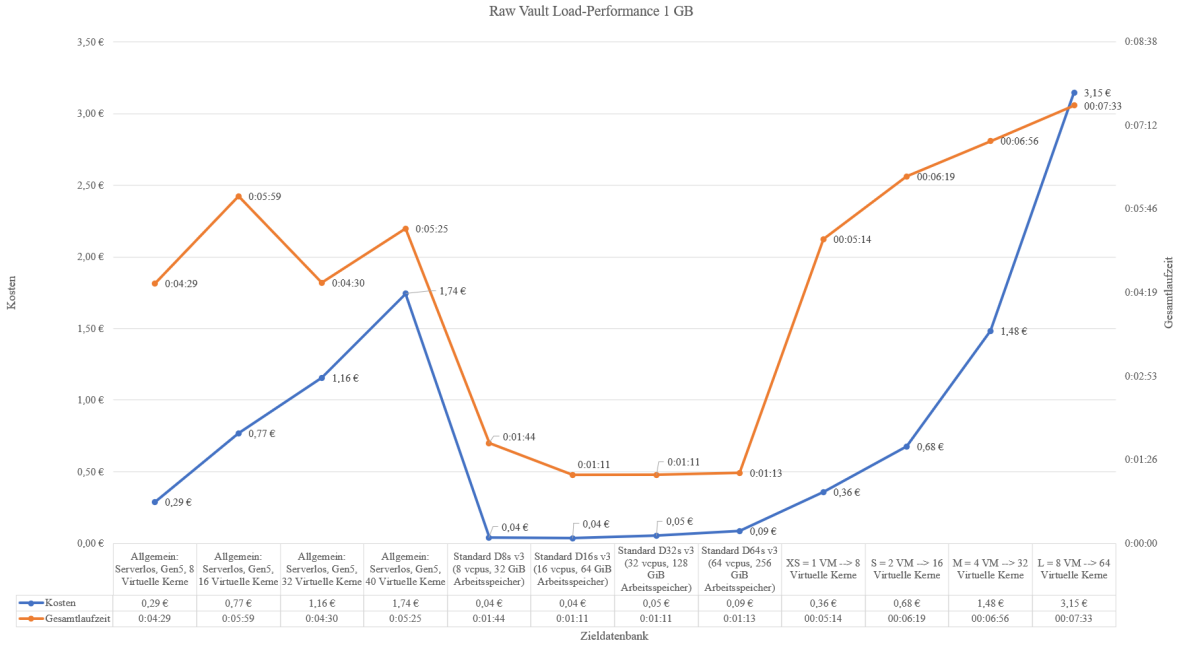


Abbildung 27: Raw Vault Load-Performance 1 GB

(Quelle: Eigene Darstellung)

Aus Abbildung 27 wird ersichtlich, dass eine höhere Skalierung der Datenbanken so gut wie keinen Einfluss auf die Gesamtlaufzeit hat. Snowflake ist im Vergleich zur Azure SQL-Datenbank und zu Exasol am langsamsten und verursacht außer in der Skalierungsstufe S die höchsten Kosten. Exasol ist deutlich schneller und günstiger als die Azure SQL-Datenbank und auch schneller sowie günstiger als Snowflake.

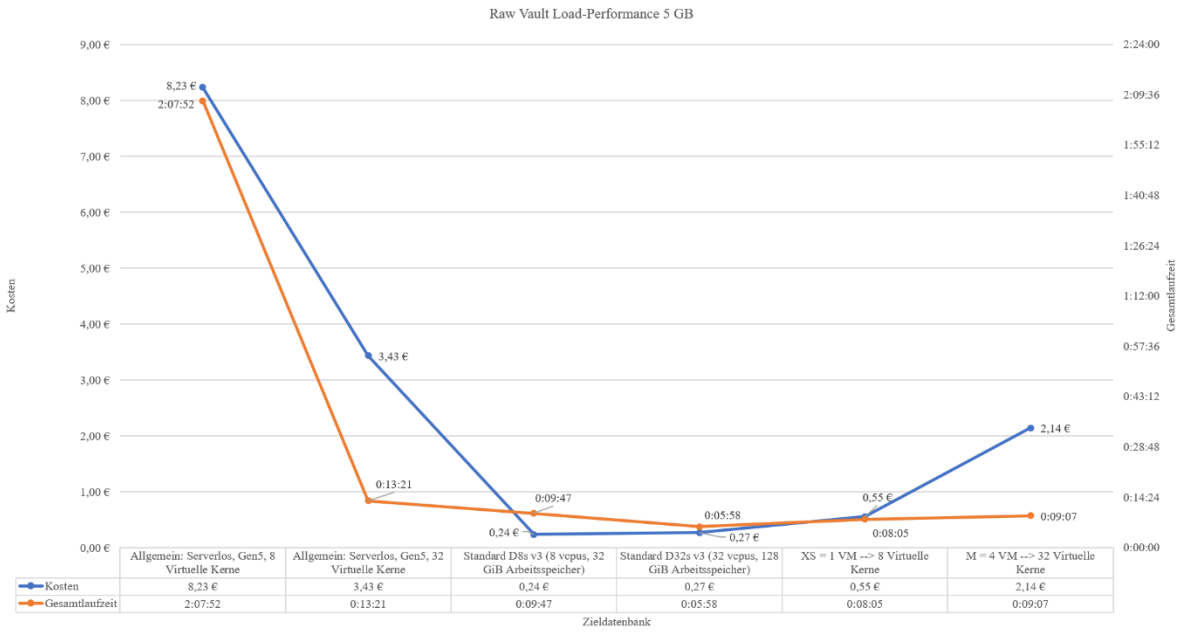


Abbildung 28: Raw Vault Load-Performance 5 GB

(Quelle: Eigene Darstellung)

Wie Abbildung 28 zu entnehmen ist, ergibt die Skalierung für die Azure SQL-Datenbank deutlich Sinn. Sie ist fast 16-mal schneller und kostet das 2,5-Fache weniger als in der kleinsten Skalierungsstufe. Auch Exasol kann durch die Skalierung die Gesamtzeit reduzieren für gerade einmal 3 Euro-Cent mehr Kosten. Für Snowflake führt die Skalierung wieder zu einer längeren Gesamtlaufzeit. Die Azure SQL-Datenbank ist im Verhältnis deutlich langsamer und teurer als Exasol und Snowflake.

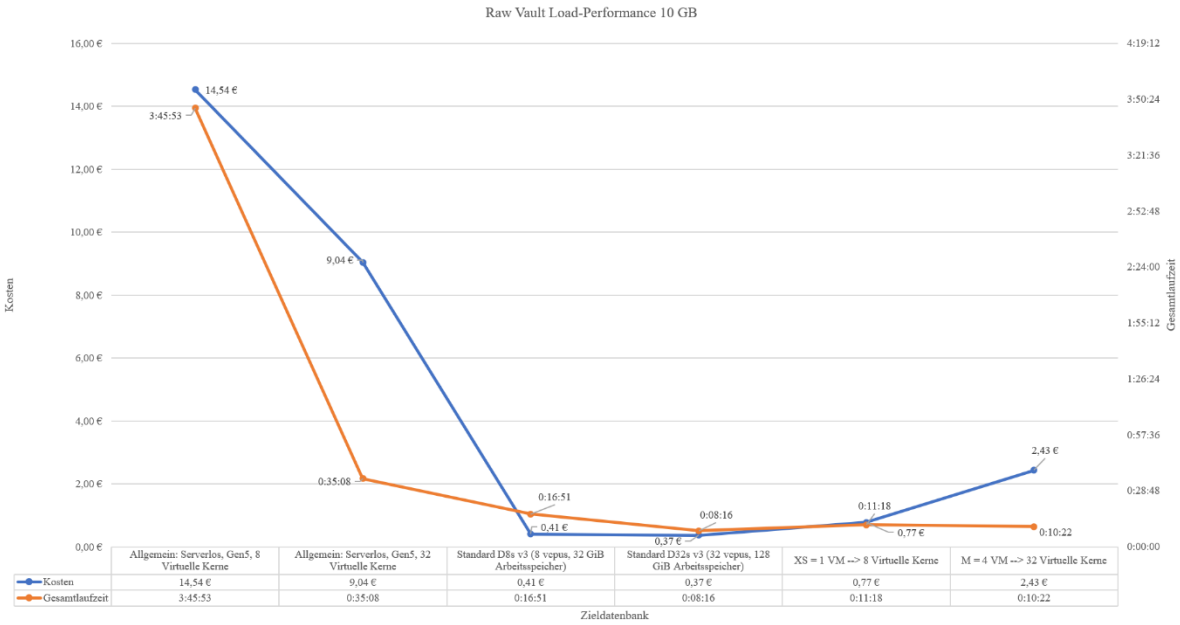


Abbildung 29: Raw Vault Load-Performance 10 GB

(Quelle: Eigene Darstellung)

Abbildung 29 zeigt, dass sich das Bild aus der Raw Vault Load-Performance für den 5 GB TPC-H-Datensatz fortzeichnet. Die Azure SQL-Datenbank profitiert, ebenso wie auch Exasol, von der Skalierung. Snowflake kann die Gesamtlaufzeit nur minimal verbessern, die Kosten steigen dafür aber um das 3-Fache an. Auch hier ist die Azure SQL-Datenbank im Verhältnis wieder deutlich teurer und langsamer als die betrachteten Alternativen.

5.1.3 Query-Performance (ohne Cache)

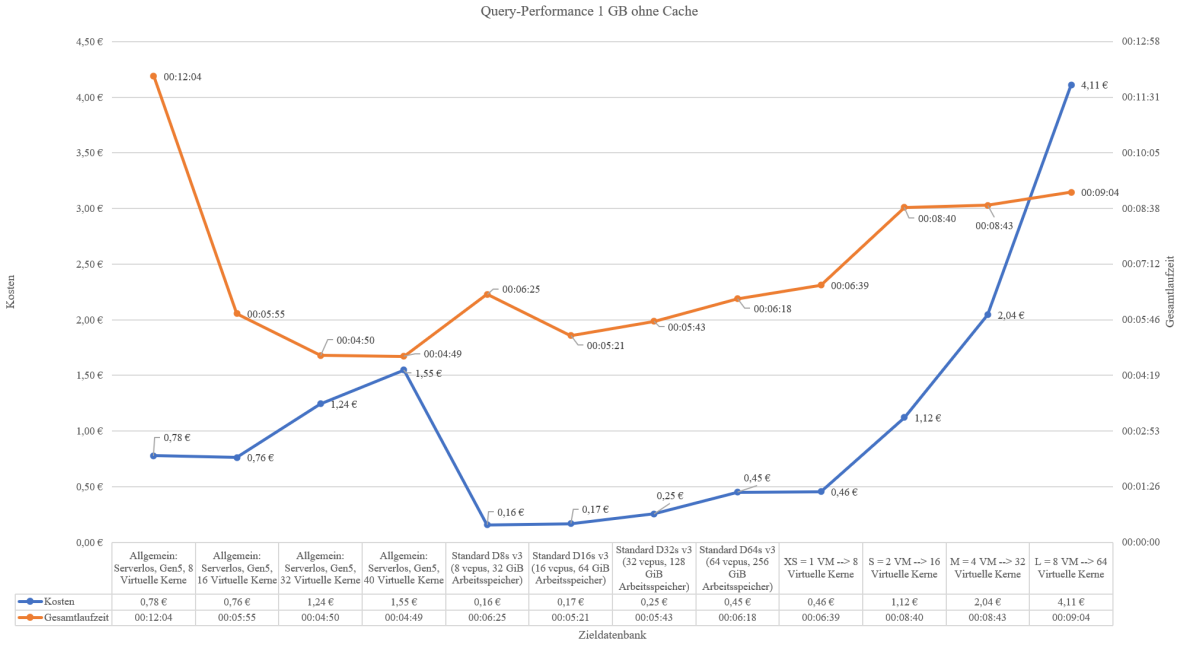


Abbildung 30: Query-Performance (ohne Cache) 1 GB

(Quelle: Eigene Darstellung)

Aus Abbildung 30 wird ersichtlich, dass eine höhere Skalierung der Datenbanken Einfluss auf die Gesamtlaufzeit der Query-Performance für die Azure SQL-Datenbank und Exasol hat. Besonders die 2. Skalierungsstufe führt für die beiden Datenbanken zu einer Verbesserung sowohl in der Gesamtlaufzeit als auch bei den Kosten. Die Azure SQL-Datenbank ist für gerade einmal 2 Euro-Cent mehr doppelt so schnell. Exasol kann die Gesamtlaufzeit für 1 Euro-Cent mehr um 1 Minute reduzieren. Snowflake hingegen profitiert nicht von einer höheren Skalierung. Die Gesamtlaufzeit sowie die Kosten steigen bei höherer Skalierung. In der 3. und 4. Skalierungsstufe ist die Azure SQL-Datenbank am schnellsten. Exasol ist im Vergleich zu den anderen beiden Datenbanken die wieder preislich günstigste.

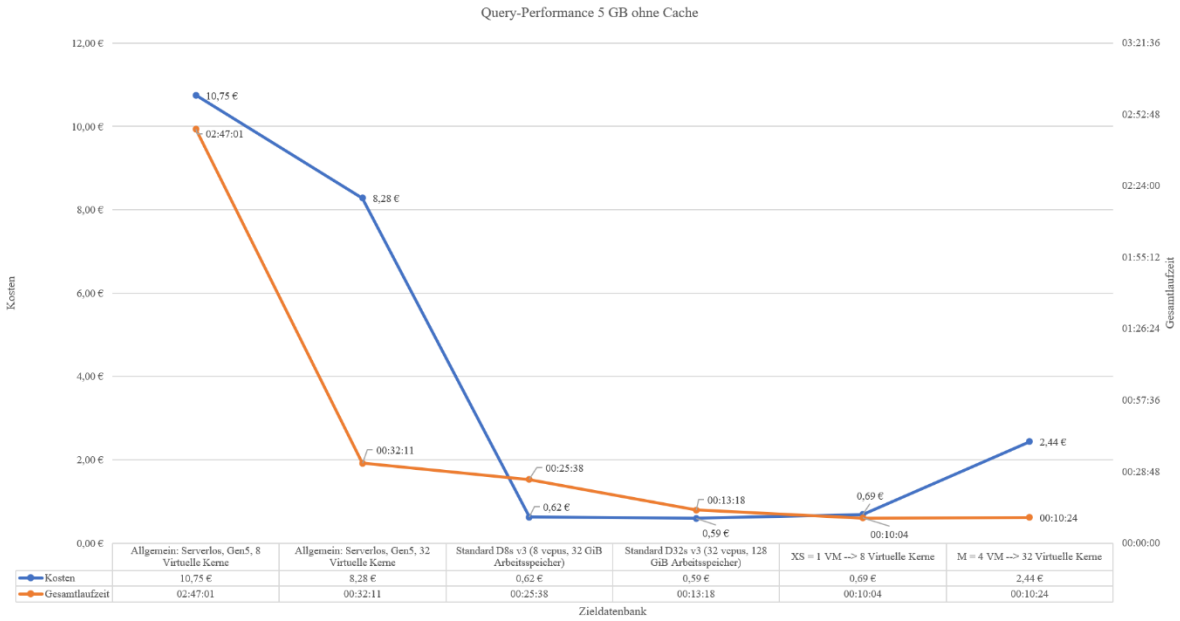


Abbildung 31: Query-Performance (ohne Cache) 5 GB

(Quelle: Eigene Darstellung)

Abbildung 31 verdeutlicht, dass eine höhere Skalierung Einfluss auf die Gesamtlaufzeit der Query-Performance für die Azure SQL-Datenbank und Exasol hat. Für Snowflake bringt die Skalierung weder eine Reduzierung der Gesamtlaufzeit noch der Kosten. Die Azure SQL-Datenbank ist mit Abstand die langsamste und von den Kosten her die teuerste. Snowflake ist in der 1. Skalierungsstufe deutlich schneller als Exasol. Preislich ist Exasol jedoch wieder am günstigsten.

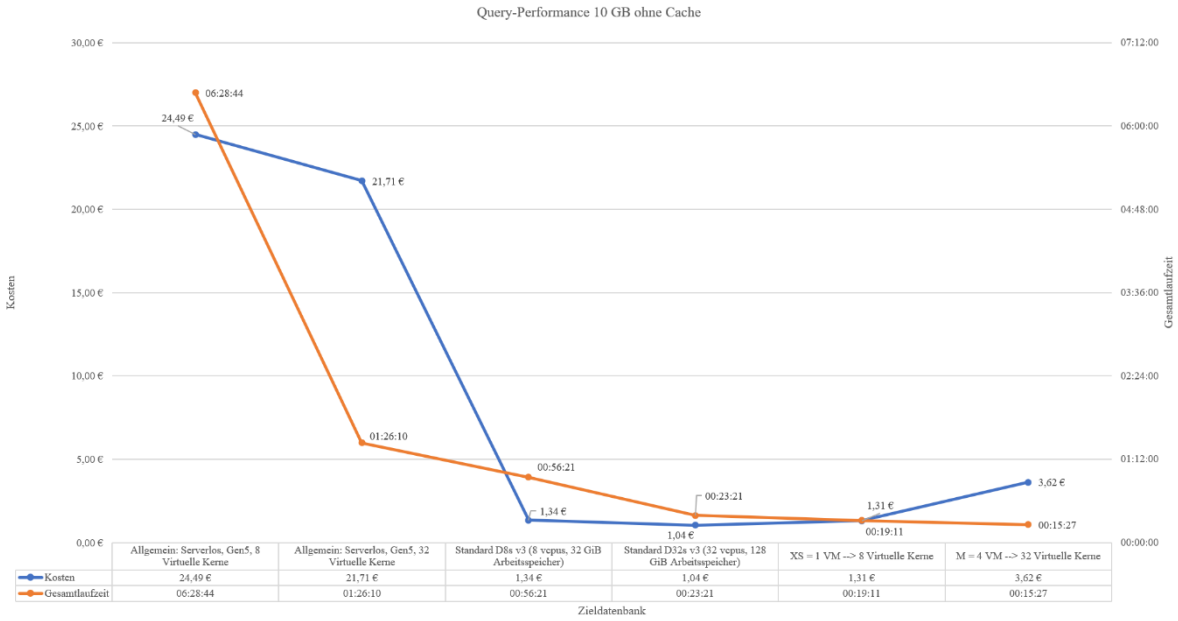


Abbildung 32: Query-Performance (ohne Cache) 10 GB

(Quelle: Eigene Darstellung)

Abbildung 32 zeigt, dass auch hier – ähnlich wie bei der Query-Performance – für den 1 und 5 GB TPC-H-Datensatz eine Skalierung für die Azure SQL-Datenbank und Exasol Sinn ergibt. Snowflake profitiert hingegen wieder nicht von der Skalierung, senkt aber erstmals die Gesamtlaufzeit mit einer höheren Skalierungsstufe. Snowflake ist, ebenso wie beim 5 GB TPC-H-Datensatz, die schnellste Datenbank. Exasol ist wieder die preislich günstigste. Die Azure SQL-Datenbank ist sowohl die langsamste als auch die teuerste Datenbank.

5.1.4 Query-Performance (mit Cache)

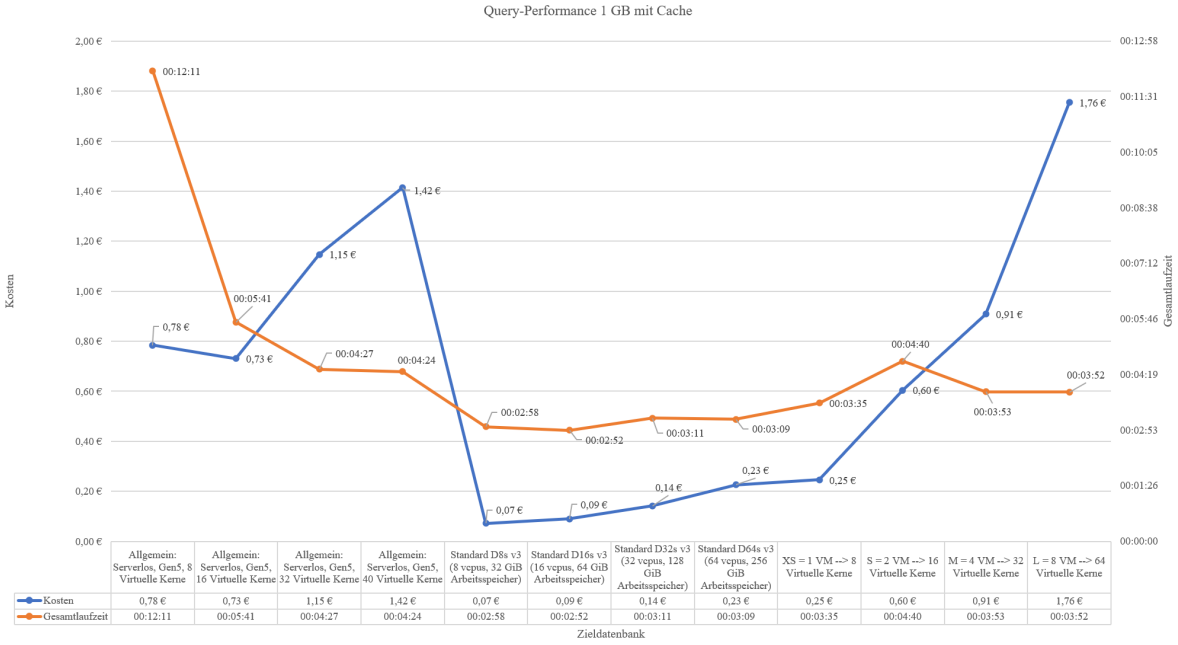


Abbildung 33: Query-Performance (mit Cache) 1 GB

(Quelle: Eigene Darstellung)

Aus Abbildung 33 wird ersichtlich, dass – mit Ausnahme der Azure SQL-Datenbank – eine höhere Skalierung der Datenbanken keinen signifikanten Einfluss auf die Gesamtlaufzeit der Query-Performance hat. Auffällig ist, dass die Gesamtlaufzeit der Azure SQL-Datenbank ähnlich der Gesamtlaufzeit der Query-Performance ohne Cache 1 GB ist. Die Azure SQL-Datenbank profitiert so gut wie gar nicht von der Nutzung des Cache. Exasol hingegen profitiert wegen seiner In-Memory-Datenbank sehr vom Cache. Auch Snowflake ist durch die Nutzung des Cache in etwa doppelt so schnell.

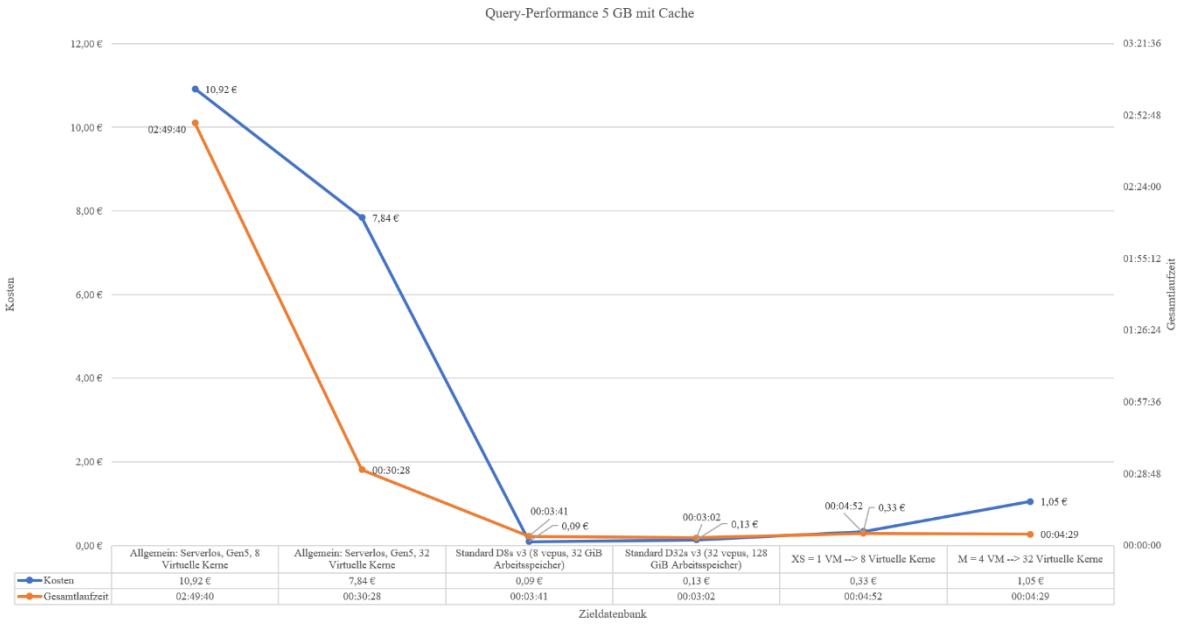


Abbildung 34: Query-Performance (mit Cache) 5 GB

(Quelle: Eigene Darstellung)

Abbildung 34 zeigt, dass auch hier die Azure SQL-Datenbank durch Nutzung des Cache nicht profitiert und ähnliche Laufzeiten wie bei der Query-Performance ohne Cache aufweist. Exasol und Snowflake profitieren hingegen von der Nutzung des Cache und senken sehr deutlich sowohl Gesamtlaufzeiten als auch Kosten.

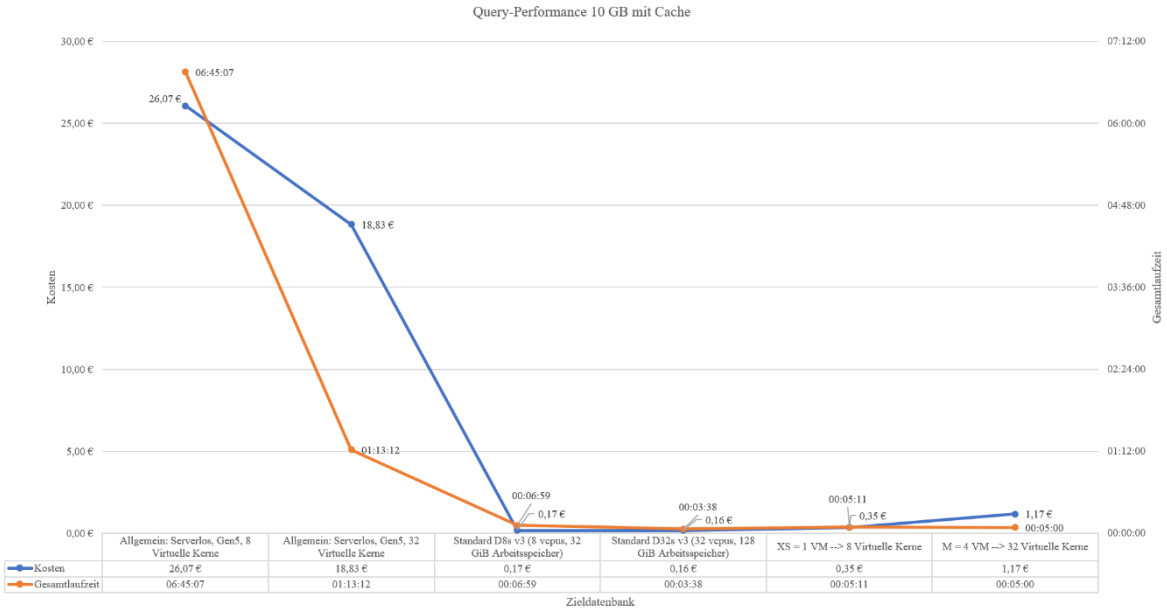


Abbildung 35: Query-Performance (mit Cache) 10 GB

(Quelle: Eigene Darstellung)

Abbildung 35 verdeutlicht, dass sich das Bild der Ergebnisse der Query-Performance mit Cache des 1 und 5 GB TPC-H-Datensatzes fortsetzt. Die Azure SQL-Datenbank weist wieder ähnliche Gesamtlaufzeiten wie bei der Query-Performance ohne Cache auf und profitiert nicht von der Nutzung des Cache. Im Gegensatz dazu profitieren Exasol und Snowflake wieder von der Nutzung des Cache. Für die 1. Skalierungsstufe weist Snowflake eine geringere Gesamtlaufzeit auf, Exasol hingegen ist wieder die preisgünstigste Datenbank.

5.1.5 Gesamtkosten und -laufzeit

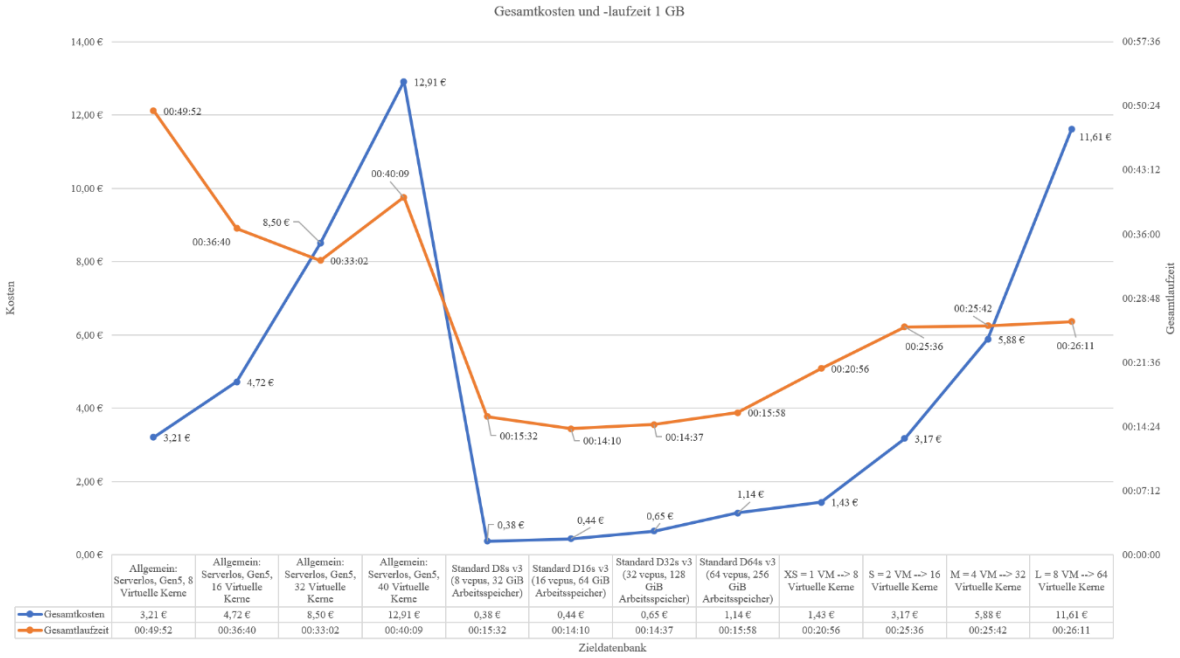


Abbildung 36: Gesamtkosten und -laufzeit 1 GB

(Quelle: Eigene Darstellung)

Abbildung 36 zeigt, dass für einen kleinen Datensatz eine Skalierung der Datenbanken keinen Sinn ergibt. Für den 1 GB TPC-H-Datensatz ist die Azure SQL-Datenbank sowohl die langsamste als auch die kostenintensivste Datenbank. Exasol hingegen ist die schnellste und günstige Datenbank. Interessant zu sehen ist, dass Snowflake in der kleinsten Skalierungsstufe deutlich schneller als in höheren Skalierungsstufen ist. Exasol ist im Verhältnis zur Azure SQL-Datenbank und Snowflake deutlich günstiger.

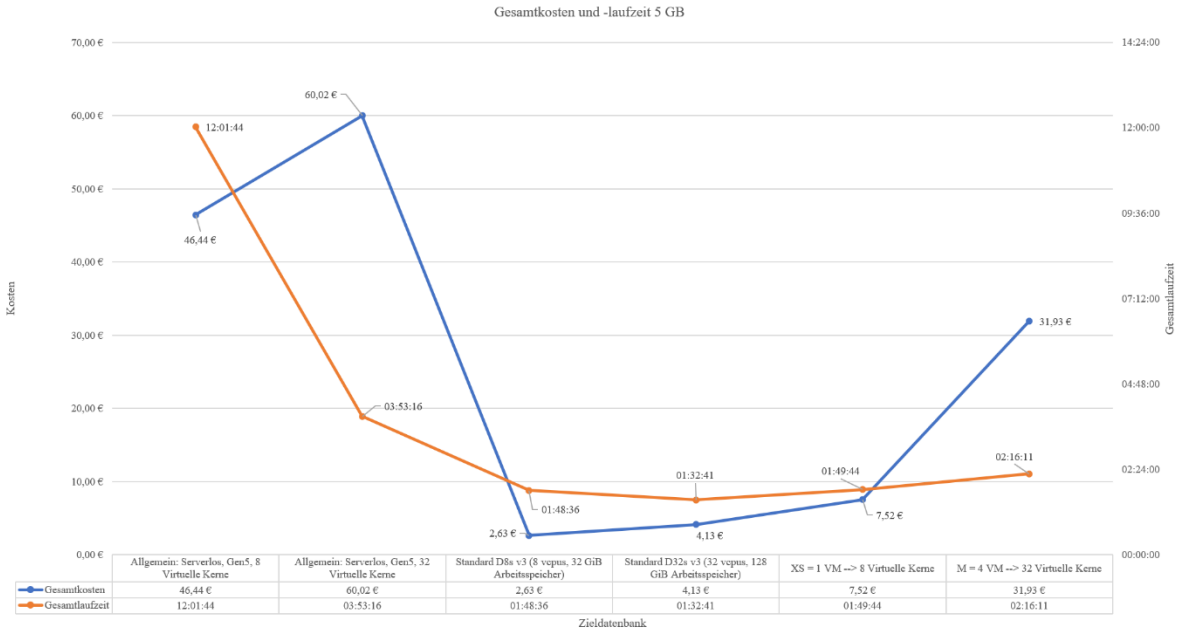


Abbildung 37: Gesamtkosten und -laufzeit 5 GB

(Quelle: Eigene Darstellung)

Abbildung 37 zeigt, dass eine Skalierung für den 5 GB TPC-H-Datensatz für die Azure SQL-Datenbank deutlich Sinn ergibt. Die Gesamtlaufzeit reduziert sich um fast 10 Stunden, wobei die Kosten dabei um lediglich 3 € steigen. Auch für Exasol ergibt die Skalierung Sinn, da die Gesamtlaufzeit reduziert werden kann und die Kosten nur minimal steigen. Für Snowflake hingeben ergibt eine Skalierung wenig Sinn. Zwar kann auch hier die Gesamtlaufzeit reduziert werden; jedoch verdreifachen sich die Kosten. Auch für den 5 GB TPC-H-Datensatz ist Exasol die günstigste und schnellste Datenbank. Die Azure SQL-Datenbank ist im Vergleich zu Exasol und Snowflake deutlich teurer und langsamer.

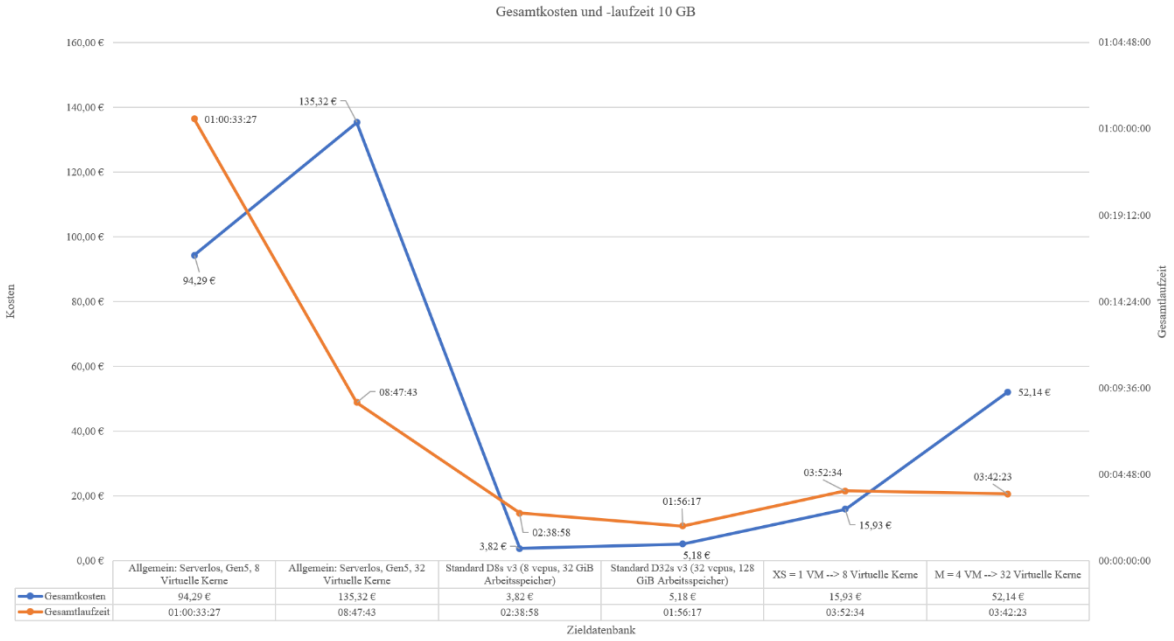


Abbildung 38: Gesamtkosten und -laufzeit 10 GB

(Quelle: Eigene Darstellung)

Aus Abbildung 38 wird deutlich, dass, ähnlich wie für den 1 und 5 GB TPC-H-Datensatz, eine Skalierung für die Azure SQL-Datenbank Sinn ergibt. Exasol ist, wie auch schon für den 1 und 5 GB TPC-H-Datensatz, die schnellste und günstigste Datenbank. Die Azure SQL-Datenbank ist, wie auch zuvor schon, deutlich teurer und langsamer als Exasol und Snowflake.

5.2 Nutzenanalyse

Für die Nutzenanalyse wird eine gemeinsame Nutzwertanalyse anhand der Vergleichsarchitektur durchgeführt. Die Betrachtung bezieht sich dabei auf die Zieldatenbanken. Hierzu wurden allgemeingültige Kriterien entwickelt. Als Entscheidungsalternativen werden die Azure SQL-Datenbank, Exasol und Snowflake verwendet.

Für die multikriterielle Bewertung der Alternativen wurden im ersten Schritt in Absprache mit der OPITZ CONSULTING GmbH und in Anlehnung an Schirm und Bensberg (vgl. Schirm und Bensberg, 2018: 86 f.) die folgenden Bewertungskriterien festgelegt (siehe dazu Tabelle 2). Solche Kriterien und ihre Relevanz lassen sich grundsätzlich aus den Unternehmenszielen beziehungsweise den Zielen der Nutzer ableiten. Im vorgestellten Fall wird die Relevanz der Kriterien ‚Administrationsaufwand' als sehr hoch, ‚Flexibilität' und ‚Skalierbarkeit' als hoch, ‚Fachwissen' als mittel und diejenige für ‚Implementierungsaufwand' sowie ‚Usability' als gering eingestuft. Es handelt sich nicht um K.-o.-Kriterien.

Nr.	Kriterium	Relevanz	Kommentar
1	Implementierungsaufwand	gering	Umfasst die Aspekte bei der Implementierung der Infrastruktur der Zielplattformen.
2	Administrationsaufwand	sehr hoch	Umfasst die anfallenden verwaltungstechnischen Belastungen.
3	Usability	gering	Umfasst die Benutzbarkeit und Benutzerfreundlichkeit der Zielplattform.
4	Flexibilität	hoch	Umfasst die Anpassbarkeit des Funktionsangebots und der Datenbasis der Lösung.
5	Skalierbarkeit	hoch	Umfasst die dynamische Anpassbarkeit der zentralen Infrastrukturressourcen.
6	Fachwissen	mittel	Umfasst das zur Verfügung stehende Fachwissen. Darunter fallen Fachbücher, Veröffentlichungen oder auch der Support der Open Source-Community.

Tabelle 2: Bewertungskriterien

(Quelle: Eigene Darstellung in Anlehnung an [Niklas, 2014: 86])

Um die Kriterien zu gewichten, wurden sie zueinander in Bezug gesetzt. Dazu wurden die Kriterien paarweise miteinander verglichen (siehe Tabelle 3). Ist ein Kriterium im Vergleich wichtiger als das andere, werden 2 Punkte vergeben. Sind die Kriterien gleich wichtig, wird 1 Punkt vergeben. Ist das Kriterium weniger wichtig, werden 0 Punkte vergeben.

In der angeführten Tabelle lautet der erste Vergleich: Der Implementierungsaufwand ist weniger wichtig als der Administrationsaufwand, also 0 Punkte. Der nächste Vergleich (1) mit (3) ergibt, dass (1) gleich wichtig ist, also 1 Punkt. Ebenso, wie es sich beim Vergleich von (1) und (2) verhalten hat, ergeben die Vergleiche (1) und (4), (1) und (5) sowie (1) und (6) 0 Punkte für das Kriterium (1).

Anzumerken ist, dass der Vergleich und damit das Ergebnis auf Annahmen des Autors hinsichtlich der Wichtigkeit beruhen; eine andere Einschätzung der Relevanz der Kriterien würde das Ergebnis entsprechend beeinflussen. Die Ergebnisse wurden spiegelbildlich eingetragen; die Summe der erzielten Werte wurde in der Spalte „Wert“ gebildet.

Kriterium		1	2	3	4	5	6	Wert
1	Implementierungsaufwand		0	1	0	0	0	1
2	Administrationsaufwand	2		2	2	2	2	10
3	Usability	1	0		0	0	0	1
4	Flexibilität	2	0	2		1	2	7
5	Skalierbarkeit	2	0	2	1		2	7
6	Fachwissen	2	0	2	0	0		4

Tabelle 3: Paarweiser Vergleich der Kriterien

(Quelle: Eigene Darstellung in Anlehnung an [Niklas, 2014])

Im vorliegenden Beispiel haben das Kriterium (2) sowie die Kriterien (4) und (5) in Relation zu den anderen Kriterien für die Entscheidungsfindung das höchste Gewicht.

Um den Gewichtungsfaktor zu bestimmen, wurden die jeweiligen Werte durch die insgesamt erzielte Wertsumme geteilt. Die Summe der Gewichtungsfaktoren ist immer 1. Die Gewichtungsfaktoren spiegeln den Anteil wider, den das Kriterium an der Gesamtentscheidung hat.

Kriterium		1	2	3	4	5	6	Wert	Faktor
1	Implementierungsaufwand		0	1	0	0	0	1	0,03
2	Administrationsaufwand	2		2	2	2	2	10	0,33
3	Usability	1	0		0	0	0	1	0,03
4	Flexibilität	2	0	2		1	2	7	0,23
5	Skalierbarkeit	2	0	2	1		2	7	0,23
6	Fachwissen	2	0	2	0	0		4	0,13
	Summe							30	1

Tabelle 4: Ermittlung der Gewichtungsfaktoren

(Quelle: Eigene Darstellung in Anlehnung an [Niklas, 2014])

Im zweiten Schritt wurde eine Skala erstellt, in der bewertet wurde, inwieweit die einzelnen Alternativen die Kriterien erfüllen.

Nr.	Kriterium	Schlechte Bewertung (0-2 Punkte)	Mittlere Bewertung (3-5 Punkte)	Gute Bewertung (6-8 Punkte)
1	Implementierungsaufwand	hoch	mittel	gering
2	Administrationsaufwand	hoch	mittel	gering
3	Usability	keine bis wenig	mittel	gut bis sehr gut
4	Flexibilität	keine bis wenig	mittel	gut bis sehr gut
5	Skalierbarkeit	keine bis wenig	mittel	gut bis sehr gut
6	Fachwissen	sehr schlecht bis schlecht	mittel	gut bis sehr gut

Tabelle 5: Bewertungsskala

(Quelle: Eigene Darstellung in Anlehnung an [Niklas, 2014])

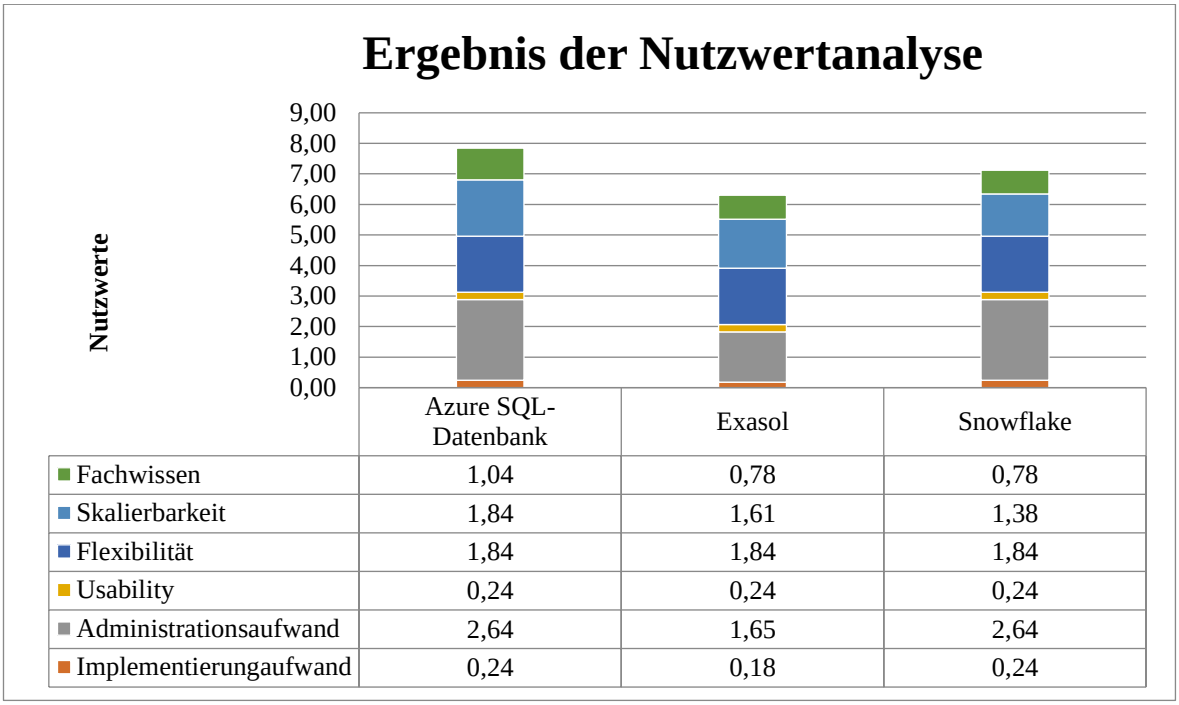
Im dritten Schritt wurden anhand der Bewertungsskala für jede Alternative Punkte vergeben. Die Punktzahl wurde in die Spalte „Bewertung nach Kriterien“ geschrieben. Der Teilnutzwert errechnete sich aus der Multiplikation des eingetragenen Punktwerts und des Gewichtungsfaktors. Die Summe der Teilnutzwerte ergab den Nutzwert der Alternative (siehe Tabelle 6).

Für den Implementierungsaufwand erhielten alle drei Lösungen hohe Punktzahlen, da der Implementierungsaufwand bei allen drei Zielplattformen sehr gering war. Im direkten Vergleich war Exasol jedoch am aufwendigsten, wodurch die Lösung minimal weniger Punkte erhält. Für den Administrationsaufwand erhielten die Azure SQL-Datenbank und Snowflake jeweils 8 Punkte. Exasol erhielt 5 Punkte. Der Punkteunterschied ist auf der Tatsache begründet, dass die Azure SQL-Datenbank eine vollständig verwaltete PaaS-Datenbank-Engine bietet, und Datenbankverwaltungsfunktionen wie beispielsweise Upgrades, Patches, Backups und Überwachung ohne Benutzerzugriff durchführt. Snowflake wird als SaaS-Lösung bereitgestellt, bei der sich Snowflake um alle Aspekte der Verwaltung kümmert. Exasol wird auf Azure über eine Azure Virtual Machine mit einem entsprechenden Image bereitgestellt. Die oben beschriebenen Datenbankverwaltungsfunktionen müssen hier selbst durchgeführt werden. Für die Usability erhielten alle drei Zielplattformen 8 Punkte, ebenso für die Flexibilität, da sie diese Kriterien vollumfänglich erfüllen. Für die Skalierung erhielt die Azure SQL-Datenbank die volle Punktzahl, da sie die feinsten Einstellungsmöglichkeiten der Skalierung bietet. Exasol erhielt 7 Punkte, da durch die Skalierung der Virtual Machine auch eine sehr feine Skalierung möglich ist. Snowflake erhielt dagegen nur 6 Punkte, da es die gröbste und restriktivste Skalierbarkeit ermöglicht. Beim Fachwissen erhielt die Azure SQL-Datenbank 8 Punkte, Exasol und Snowflake wurden mit jeweils 6 Punkten bewertet.

Bewertung der Alternativen		**GF**	**Bewertung nach Kriterium**	**Teilnutzwert**	**Bewertung nach Kriterium**	**Teilnutzwert**	**Bewertung nach Kriterium**	**Teilnutzwert**
Nr.	Kriterium		Azure SQL-Datenbank		Exasol		Snowflake	
1	Implementierungsaufwand	0,03	8	0,24	6	0,18	8	0,24
2	Administrationsaufwand	0,33	8	2,64	5	1,65	8	2,64
3	Usability	0,03	8	0,24	8	0,24	8	0,24
4	Flexibilität	0,23	8	1,84	8	1,84	8	1,84
5	Skalierbarkeit	0,23	8	1,84	7	1,61	6	1,38
6	Fachwissen	0,13	8	1,04	6	0,78	6	0,78
		Ergebnis	48	7,84	40	6,30	44	7,12

Tabelle 6: Ergebnis der Nutzwertanalyse

(Quelle: Eigene Darstellung in Anlehnung an [Niklas, 2014])



	Azure SQL-Datenbank	Exasol	Snowflake
Fachwissen	1,04	0,78	0,78
Skalierbarkeit	1,84	1,61	1,38
Flexibilität	1,84	1,84	1,84
Usability	0,24	0,24	0,24
Administrationsaufwand	2,64	1,65	2,64
Implementierungaufwand	0,24	0,18	0,24

Abbildung 39: Ergebnis der Nutzwertanalyse

(Quelle: Eigene Darstellung in Anlehnung an [Niklas, 2014])

6 Gesamtbewertung

Im sich nun anschließenden Kapitel wird die Bewertung, basierend auf den zuvor dargestellten Performancemessungen und Nutzen der Zielplattformen Azure SQL-Datenbank, Exasol und Snowflake am Beispiel des DVB zur automatischen DWH-Erstellung in der Azure Cloud vorgenommen. Auf die Ergebnisse wird nachfolgend detailliert und über die einfache Kommentierung der Diagramme in den Abschnitten 5.1.1 bis 5.1.5 hinausgehend eingegangen.

Bei der Betrachtung der Ingest-Performance für den 1, 5 und 10 GB TPC-H-Datensatz ist ein deutlicher Vorteil in der Gesamtlaufzeit und den Kosten bei Exasol zu erkennen. Dieser spiegelt sich in der Differenz der Gesamtlaufzeit und den Kosten zugunsten von Exasol wider.

Ein ähnliches Ergebnis zeigte sich bei der Betrachtung der Raw Vault Load-Performance. Auch hier ist ein deutlicher Vorteil in der Gesamtlaufzeit und den Kosten für den 1, 5 und 10 GB TPC-H-Datensatz für Exasol zu erkennen. Dies ergibt sich wieder durch die Differenz der Gesamtlaufzeit und Kosten für die unterschiedlichen Skalierungsstufen und Datensatzgrößen.

Für die Query-Performance ohne Cache ist für den 1 GB TPC-H-Datensatz die Gesamtlaufzeit für die 3. und 4. Skalierungsstufe der Azure SQL-Datenbank geringer als die von Exasol und Snowflake. Die Unterschiede sind nur marginal und rechtfertigen die dadurch

entstehenden höheren Kosten nicht. Für den 5 und 10 GB TPC-H-Datensatz haben sich die Ergebnisse umgekehrt: Snowflake weist die geringste Gesamtlaufzeit auf. Bei der Betrachtung der Kosten liegt Exasol jeweils unter den von der Azure SQL-Datenbank und Snowflake für den 1, 5 und 10 GB TPC-H-Datensatz.

Für die Query-Performance mit Cache für den 1, 5 und 10 GB TPC-H-Datensatz ist wieder ein deutlicher Vorteil in der Gesamtlaufzeit und bei den Kosten bei Exasol zu erkennen. Die Azure SQL-Datenbank profitiert durch die Nutzung des Cache so gut wie gar nicht und hat ähnliche Gesamtlaufzeiten und Kosten wie bei der Query-Performance ohne Cache. Snowflake profitiert zwar von der Nutzung des Cache, kommt aber nicht an die Gesamtlaufzeiten von Exasol heran. Exasol profitiert hierbei hingegen außerordentlich von seiner In-Memory-Technologie. Auch bei der Query-Performance mit Cache liegt Exasol bei der Betrachtung der Kosten jeweils unter denen der Azure SQL-Datenbank und Snowflake für den 1, 5 und 10 GB TPC-H-Datensatz.

Bei der kumulierten Gesamtlaufzeit und Kostendarstellung im Vergleich der Alternativen ist zu erkennen, dass die Gesamtlaufzeit und Kosten für Exasol diejenigen der Azure SQL-Datenbank und Snowflake bereits für den 1 GB-Datensatz unterschreiten. Je größer der Datensatz wird, desto deutlicher sticht die Vorteilhaftigkeit von Exasol heraus. Festzustellen ist, dass, je größer der Datensatz wird (gut ersichtlich beim 10 GB TPC-H-Datensatz), Snowflake deutlich schneller als Exasol in der Query-Performance ohne Cache ist.

Weiter ist festzustellen, dass eine Skalierung der Datenbanken für den 1 GB TPC-H-Datensatz keinen Sinn ergibt, da sich die Kosten im Verhältnis zur gewonnenen Gesamtlaufzeit zu stark erhöhen. Ab dem 5 GB-TPC-H-Datensatz hat eine Skalierung der Azure SQL-Datenbank sowie von Exasol durchaus Sinn ergeben. Dabei konnten die Gesamtlaufzeiten und teilweise auch die Kosten signifikant gesenkt werden. Für Snowflake allerdings hat eine Skalierung sich nicht als sinnvoll erwiesen. Betrachtet man die Gesamtlaufzeit und -kosten der jeweiligen TPC-H-Datensätze, so ist recht schnell zu erkennen, dass die minimal kürzere Gesamtlaufzeit bei höheren Skalierungsstufen in keinem guten Verhältnis zu den gestiegenen Kosten steht. Für den 1 GB TPC-H-Datensatz ist die kleinste Skalierungsstufe von Snowflake sogar die performanteste.

Bei der Nutzwertanalyse erhielt Exasol die schlechteste Bewertung. Die Datenbanklösung erzielte einen Nutzwert von 6,30, wohingegen Snowflake einen Nutzwert von 7,12 und die Azure SQL-Datenbank einen Nutzwert von 7,84 erzielte. Die schlechtere Bewertung ist zum einen auf den höheren Administrationsaufwand und zum anderen auf den höheren

Implementierungsaufwand zurückzuführen. Bei Snowflake erfolgt durch die gröbste und restriktivste Skalierungsmöglichkeit eine schlechtere Bewertung als bei den beiden anderen Lösungen. Gerade in diesem Zusammenhang der Bewertungen sei allerdings nochmals auf den subjektiven Charakter der hier vorgestellten Nutzwertanalyse hingewiesen; insbesondere sei betont, dass die Kriterienauswahl und Kriteriengewichtung für andere Unternehmen anders aussehen kann.

Für alle drei TPC-H-Datensatzgrößen stellt sich Exasol trotz des schlechteren Nutzwertes als vorteilhafter heraus: zum einem deshalb, weil die Datenbanklösung in jeder Skalierungsstufe die geringste Gesamtlaufzeit gegenüber der Azure SQL-Datenbank und Snowflake erzielt hat; zum anderen aus dem Grund, weil die Datenbanklösung nur einen Bruchteil der Kosten der anderen beiden Zieldatenbanken verursacht.

Die Azure SQL-Datenbank ist aufgrund ihrer langen Laufzeiten und den damit verbunden Kosten kaum für den Einsatz mit dem DVB zu empfehlen. Snowflake hingegen ist nicht unmittelbar viel langsamer als Exasol, verursacht aber erheblich mehr Kosten. Der Vorteil, den Snowflake jedoch gegenüber Exasol aufweist, ist darin zu sehen, dass Snowflake als SaaS-Lösung bereitgestellt wird, bei der Snowflake selbst alle Aspekte der Verwaltung abdeckt. Exasol hingegen wird auf Azure über eine Azure Virtual Machine mit einem entsprechenden Image bereitgestellt: Upgrades, Patches, Backups und Überwachung müssen hier eigenständig vom Nutzer durchgeführt werden. Inwieweit dies zu weiteren Kosten und Aufwand bei der Nutzung von Exasol führt bzw. führen kann, muss je nach individueller Ausgangslage geprüft und kalkuliert werden. Grundsätzlich bieten sowohl Exasol als auch Snowflake unterschiedliche Performance-/ Kosten-/ Nutzen-Vorteile, die je nach individueller Situation zielführender für ein Unternehmen sein können.

Generell ist allerdings anzumerken, dass bei den vorliegenden Performanceergebnissen kein Datenbank-Tuning stattgefunden hat. Die Feinabstimmung von Parametern einer Datenbank oder die Optimierung des Datenmodells im Allgemeinen könnten zu einer Performance-Verbesserung und möglicherweise zu grundlegend anderen Ergebnissen führen.

Die eingangs (siehe Abschnitt 1.2) gestellten Fragen können nunmehr aufgrund der erzielten Ergebnisse wie folgt beantwortet werden:

Zu Frage 1, in welcher Kombination der DVB am performantesten ist: Der DVB ist in Kombination mit Exasol als Zielplattform für alle drei getesteten TPC-H-Datensätze am performantesten. Für den 5 und 10 GB TPC-Datensatz ist jedoch Snowflake bei der Query-Performance ohne Cache performanter.

Zu Frage 2, in welcher Kombination der DVB am einfachsten zu handhaben ist: Die Handhabung des DVB ist mit allen drei Zielplattformen einfach und gut. Bis auf eine anfängliche Installation von Objekten, hauptsächlich Tabellen für Konfiguration und Protokollierung sowie Metadatenansichten, sowie beim Updaten des DVB, unterscheiden sich die Zieldatenbanken in der Handhabung nicht.

Zu Frage 3, in welcher Kombination der DVB preislich am günstigsten ist: Durch die in der Arbeit vorgestellte Performanceanalyse auf unterschiedlich großen TPC-H-Datensätzen wurde klar ersichtlich, dass eine Lösung günstiger ist. Es zeigte sich, das Exasol in allen Skalierungsstufen für alle untersuchten TPC-H-Datensatzgrößen zum günstigsten Ergebnis führte.

Zu Frage 4, welchen Nutzen die unterschiedlichen Zieldatenplattformen bieten: Der Nutzen, den die Azure SQL-Datenbank bietet, besteht aus:

- einer vollständig verwalteten PaaS-Datenbank,
- einer Verfügbarkeit von 99,99 %,
- aus verschiedenen Datenbankbereitstellungsoptionen,
- der Skalierbarkeit,
- unterschiedlichen Kaufmodellen

  sowie:
- unterschiedlichen Serviceebenen.

Der Nutzen, den Exasol bietet, besteht aus:

- In-Memory-Technologie,
- spaltenbasierter Speicherung und Komprimierung,
- massiv paralleler Verarbeitung (MPP),
- einer hohen Benutzerparallelität,
- Skalierbarkeit,
- Selbstoptimierung

  sowie:
- flexiblen Bereitstellungsoptionen.

Der Nutzen, den Snowflake bietet, besteht aus:

- einer vollständig verwalteten SaaS-Lösung,
- einer einfachen Implementierung,
- Skalierbarkeit,

- einer hybriden Datenbankarchitektur aus Shared Disk- und Shared Nothing,
- einer intelligenten Datenspeicherung
  sowie
- wöchentlichen Serviceverbesserungen.

Zu Frage 5, ob weitere Anbindungsmöglichkeiten beispielsweise zu Azure Analysis Services und/oder Power BI bestehen: Für alle drei Zielplattformen bestehen weitere Anbindungsmöglichkeiten, explizit auch zu Azure Analysis Services und/oder Power BI.

7 Zusammenfassung und Fazit

In der vorliegenden Arbeit wurden mit Hilfe einer Performance-Nutzen-Analyse verschiedene Zielplattformen in Kombination mit dem DVB zur automatischen DWH-Erstellung verglichen, um eine Aussage über die Vorteilhaftigkeit einer Zielplattform für den DVB im Hinblick auf die Performance und Nutzung geben zu können. Zudem soll die Kenntnis der Arbeit es ermöglicht haben, die Thematik der DWHA tiefer zu durchdringen, um die jeweils bestmögliche Nutzung in Kundenprojekten zu garantieren. Zunächst wurden die notwendigen informationstechnischen Grundlagen behandelt und die verwendeten Technologien und Werkzeuge beschrieben, die zur Gegenüberstellung der Performance- und Nutzenanalyse herangezogen wurden. Die Untersuchung erfolgte in Form einer Fallstudie. Im Anschluss daran wurde auf die verwendeten Methoden eingegangen, der Benchmark erläutert; es wurden die Rahmenbedingungen der Performance-Nutzen-Analyse festgelegt und die Vergleichsarchitektur vorgestellt. Da einerseits eine ausschließliche Performancebetrachtung als zu einseitig anzusehen ist und andererseits eine rein qualitative Vorgehensweise dem Rationalitätsprinzip der Wirtschaftlichkeit widerspricht, wurde eine Performance-Nutzen-Analyse durchgeführt. Abschließend wurden die Ergebnisse dieser Performance-Nutzen-Analyse bewertet.

Die Ergebnisse der Performance-Nutzen-Analyse belegen, dass Exasol sowohl performanter als auch wirtschaftlicher für alle untersuchten TPC-H-Datensatzgrößen ist. Dennoch lässt sich daraus nicht definitiv ableiten, das Exasol in jedem Fall die wirtschaftlichere Lösung darstellt, da die Bereitstellungsform von Exasol zu weiteren Aufwänden und Kosten seitens des Herstellers führen kann, die in dieser Arbeit jedoch keine Betrachtung gefunden haben. Snowflake ist zusätzlich zu Exasol in Betracht zu ziehen, wenn man sich nicht explizit mit Datenbankverwaltungsfunktionen befassen möchte. Die Azure SQL-Datenbank ist aufgrund der erzielten Ergebnisse nicht zu empfehlen.

Festzustellen ist, dass eine Skalierung für den 1 GB TPC-H-Datensatz keinen Mehrwert hinsichtlich der Laufzeit gebracht hat – im Gegenteil. Die Laufzeiten für Exasol und Snowflake sind bei höheren Skalierungen sogar gestiegen. Für den 5 und 10 GB TPC-H-Datensatz hat eine Skalierung für die Azure SQL-Datenbank und Exasol zu Verbesserung in der Gesamtlaufzeit und teilweise in den Kosten geführt. Snowflake hingegen hat über alle drei TPC-H-Datensatzgrößen von einer Skalierung nicht profitiert. Interessant wäre ein Vergleich der Zieldatenbanken mit unterschiedlichen Skalierungsstufen für TPC-H-Datensatzgrößen >= 10 TB oder generell ein Vergleich weiterer Zieldatenbanken wie beispielsweise Azure Synapse Analytics.

Zusammenfassend lässt sich festhalten, dass grundsätzlich Exasol und Snowflake zu nennen sind bei der Frage, in welcher Kombination der DVB im Hinblick auf Performance und Nutzung vorteilhafter ist.

Abschließend wird darauf hingewiesen, dass aufgrund des dynamischen Charakters der IT-Branche in kurzen Zeitabständen signifikante Veränderungen innerhalb des Untersuchungsfeldes möglich sind, da sich beispielsweise Services im Cloud-Umfeld stets ändern können. Die Schnelllebigkeit kann bereits erzielte Ergebnisse rasch obsolet werden lassen und neue Forschungsaktivitäten initiieren. Vor diesem Hintergrund ist eine Überprüfung der Performance der Zieldatenbank bei Einführung einer neuen DVB-Lösung vorzunehmen – auch beziehungsweise nicht zuletzt in Bezug auf die in der vorliegenden Arbeit betrachteten Zieldatenbanken. Wie aber die Ergebnisse aus der vorliegenden Untersuchung zeigen, ist eine solche Anforderung zu jedem Zeitpunkt als lohnend und insofern als im Wortsinn wertvoll zu erachten.

Literaturverzeichnis

2150 GmbH - Features Overview, in: Datavault-builder (Stichwort). o. J. [Online] o. J. [Zitat vom: 23. Juli 2021.] https://datavault-builder.com/de/features/.

2150 GmbH - Home, in: datavault-builder (Stichwort). o. J. [Online] o. J. [Zitat vom: 2021. Juli 23.] https://datavault-builder.com/de/.

Balaji, Sundramoorthy und Murugaiyan, Sundararajan. 2012. WATEERFALLVs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC. International Journal of Information Technology and Business Management (JITBM). 2012, Vol. 2, No. 1, S. 26-30.

Barata, Melyssa, Bernardino, Jorge und Furtado, Pedro. 2015. An Overview of Decision Support Benchmarks: TPC-DS, TPC-H and SSB. [Hrsg.] Alvaro Rocha, et al. New Contributions in Information Systems and Technologies. Vol. 1. Cham: Springer International Publishing, 2015, S. 619-628.

Bauer, Andreas und Günzel, Holger. 2013. Data Warehouse Systeme: Architektur, Entwicklung, Anwendung. 4., überarbeitete und erweiterte Auflage. Heidelberg: dpunkt.verlag, 2013.

Blaschek, Klaus. 2017. Die sechs Facetten der BI/DWH-Automatisierung Schneller, effizienter und qualitativ besser, in: ONLINE-THEMENSPECIAL DATA WAREHOUSE AUTOMATION. [Online] 2017. [Zitat vom: 3. Juni 2021.] http://www.sigs.de/publications/bi/2017/DataWarehouseAutomation/blaschek_BIS_OTS_DWA_2017.pdf.

Blohm, Hans, Lüder, Klaus und Schaefer, Christina. 2013. Investition: Schwachstellenanalyse des Investitionsbereichs und Investitionsrechnung. 10., bearbeitete und aktualisierte Auflage. München: Franz Vahlen, 2013.

Branger, Raphael. 2017. Agile BI in der Praxis: Data Warehouse Automation, in: ONLINE-THEMENSPECIAL DATA WAREHOUSE AUTOMATION. [Online] 2017. [Zitat vom: 8. Juli 2021.] http://www.sigs.de/publications/bi/2017/DataWarehouseAutomation/branger_BIS_OTS_DWA_2017.pdf.

Brooks, Frederick P. 1975. The Mythical Ma-month: Essays on Software Engineering. Chapel Hill, N.C.: Addison-Wesley, 1975.

CSV Logo in: Flaticon (Stichwort). o. J. [Online] o. J. [Zitat vom: 23. Juli 2021.] https://image.flaticon.com/icons/png/512/28/28842.png.

Datavault Builder – Ihr Data Warehouse Automatisierungs-Werkzeug der 4. Generation, in: Xax (Stichwort). o. J. [Online] o. J. [Zitat vom: 12. Juli 2021.] https://www.xax.de/software/datavault-builder/.

Devlin, Barry. 2015. BI, Built to Order, On-Demand Automating Data Warehouse Delivery. [Online] Januar 2015. [Zitat vom: 7. Juli 2021.] http://9sight.com/pdfs/Built_to_Order_White_Paper.pdf.

Docker Logo, in: Testautomatisierung (Stichwort). o. J. [Online] o. J. [Zitat vom: 23. Juli 2021.] https://www.testautomatisierung.org/lexikon/docker/14098888813_bec60d595d_o/.

Docker overview, in: Docker (Stichwort). o. J. [Online] o. J. [Zitat vom: 10. Juli 2021.] https://docs.docker.com/get-started/overview/.

Dremio. 2021. What is a Data Warehouse?, in Dremio. [Online] 2021. [Zitat vom: 1. Juli 2021.] https://www.dremio.com/data-lake/data-warehouse/.

Duvuri, Aditya. 2020. Slowly Changing Dimensions (SCD) Type 2 Implementation in Oracle Cloud Infrastructure (OCI) Data Integration. [Online] 12. November 2020. [Zitat vom: 16. Juni 2021.] https://blogs.oracle.com/dataintegration/slowly-changing-dimensions-scd-type-2-implementation-in-oracle-cloud-infrastructure-oci-data-integration.

Eckerson, Wayne. 2015. Data Warehouse Automation Tools: Product Categories and Positioning. [Online] September 2015. [Zitat vom: 3. Juni 2021.] https://www.eckerson.com/register?content=which-data-warehouse-automation-tool-is-right-for-you.

Evelson, Boris und Angel, Nasry. 2016. Data Warehouse Automation Helps Close The Data-To-Insight-Gap. [Online] 16. August 2016. [Zitat vom: 3. Juni 2021.] https://www.wherescape.com/media/2941/data_warehouse_automation_helps_close_the_data-to-insight_gap.pdf.

Exasol - die hoch performante In-Memory-Analytics-Datenbank, in: Areto (Stichwort). o. J. [Online] o. J. [Zitat vom: 13. Juli 2021.] https://www.areto.de/technologien/data-warehouse/exasol/.

Exasol Overview, in: Exasol (Stichwort). o. J. [Online] o. J. [Zitat vom: 13. Juli 2021.] https://docs.exasol.com/get_started/exasol_overview.htm.

Exasol Vector Logo in: Seekvectorlogo (Stichwort). 2021. [Online] 13. April 2021. [Zitat vom: 23. Juli 2021.] https://seekvectorlogo.com/exasol-vector-logo-svg/.

Farkisch, Kiumars. 2011. Data-Warehouse-Systeme kompakt Aufbau, Architektur, Grundfunktion. Berlin: Springer, 2011.

Fiore, Vincenzo. 2018. Alles auf Anfang - Digitalisierung ausprobieren, in: Computerwoche. [Online] 18. Juli 2018. [Zitat vom: 12. Juli 2021.] https://www.computerwoche.de/a/alles-auf-anfang-digitalisierung-ausprobieren,3545398.

Franke, Kurt. 2008. Daten-Historisierung im Data Warehouse. [Online] April 2008. [Zitat vom: 16. Juni 2021.] https://www.doag.org/formes/pubfiles/304099/docs/Publikationen/DOAGNews/2008/2008-4/2008-04-News-Franke-Daten_Historisierung.pdf.

Goeken, Matthias. 2006. Entwicklung von Data-Warehouse-Systemen: Anforderungsmanagement, Modellierung, Implementierung. Wiesbaden : Deutscher Universitäts-Verlag / GWV Fachverlage GmbH, 2006.

Götze, Uwe. 2014. Investitionsrechnung: Modelle und Analysen zur Beurteilung. 7. Auflage. Berlin/Heidelberg: Springer-Verlag, 2014.

Gruschka, Herbert und Schnitzenbaumer, Franz. 2008. Methoden zu Wirtschaftlichkeitsbetrachtungen bei Investitionsmaßnahmen und Angeboten, in: Bayerischer Kommunaler Prüfungsverband - Geschäftsbericht 2008. München: Bayerischer Kommunaler Prüfungsverband, 2008. S. 78-94.

Hahne, Michael. 2014. Modellierung von Business-Intelligence-Systemen: Leitfaden für erfolgreiche Projekte auf Basis flexibler Data-Warehouse-Architekturen. Heidelberg: dpunkt.verlag, 2014.

Heinrich, Lutz J., Heinzl, Armin und Roithmayr, Friedrich. 1998. Performance (Stichwort), in: Wirtschaftsinformatik-Lexikon. 6., vollständig überarbeitete und erweiterte Auflage. München: Oldenbourg, 1998.

Inmon, William H. 2005. Building the Data Warehouse. 4. Auflage. Indianpolis: Wiley Publishing, 2005.

Inmon, William H., Linstedt, Daniel und Levins, Mary. 2019. Data Architecture: A Primer for the Data Scientist. 2. Auflage. San Diego: Elsevier Science & Technolog, 2019.

Inmon, William H., Strauss, Derek und Neushloss, Genia. 2008. DW 2.0: The Architecture for the Next Generation of Data Warehousing. Amsterdam: Elsevier, Morgan Kaufmann, 2008.

Irish, Paul. 2010. Highres Browser Logos, in: Paulirish. [Online] 14. August 2010. [Zitat vom: 23. Juli 2021.] https://www.paulirish.com/2010/high-res-browser-icons/.

Key Concepts & Architecture, in: Snowflake (Stichwort). o. J. [Online] o. J. [Zitat vom: 13. Juli 2021.] https://docs.snowflake.com/en/user-guide/intro-key-concepts.html.

Kimball, Ralph, et al. 2011. The Data Warehouse Lifecycle Toolkit: Practical Techniques for Building Data Warehouse and Business Intelligence Systems. 2. Auflage. New York, NY: John Wiley & Sons, 2011.

Kimball, Ralph und Ross, Margy. 2013. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. 3. Auflage. New York, NY: John Wiley & Sons, 2013.

Köppen, Veit, Sattler, Kai-Uwe und Saake, Gunter. 2014. Data Warehouse : Technologien. 2. Auflage. Heidelberg: mitp, 2014.

Lans, Rick F. van der. 2012. Data Virtualization for Business Intelligence Systems: Revolutionizing Data Integration for Data Warehouses. Waltham, MA: Morgan Kaufmann, 2012.

Lehner, Wolfgang. 2003. Datenbanktechnologie für Data-Warehouse-Systeme Konzepte und Methoden. Heidelberg: dpunkt.verlag, 2003.

Linstedt, Daniel und Olschimke, Michael. 2016. Building a scalable data warehouse with Data Vault 2.0. Amsterdam: Elsevier, 2016.

Luber, Stefan und Litzel, Nico. 2018. Stichwort: ETL, in: Bigdata-insider. [Online] 16. November 2018. [Zitat vom: 8. Juni 2021.] https://www.bigdata-insider.de/was-ist-etl-extract-transform-load-a-776549/.

Lücke, Wolfgang,[Hrsg.]. 1991. Nutzwertanalyse (Stichwort), in: Investitionslexikon . 2., völlig neubearbeitete und erweiterte Auflage. München: Vahlen-Verlag, 1991.

Microsoft. 2017. Enterprise Data Warehouses and BI in the Age of Cloud Computing and Big Data Six illustrated use cases. [Online] 2017. [Zitat vom: 10. Juli 2021.] https://www.justanalytics.com/hubfs/JA%20-%20Home%20page%20-%20ebooks,%20white%20papers/Enterprise%20Data%20Warehouses%20and%20BI%20in%20the%20Age%20of%20Cloud%20Computing%20and%20big%20Data.pdf.

Niklas, Cornelia. 2014. Mehr Entscheidungssicherheit mit der Nutzwertanalyse, in: Projektmagazin. [Online] Juli 2014. [Zitat vom: 24. Juli 2021.] https://www.projektmagazin.de/artikel/mehr-entscheidungssicherheit-mit-der-nutzwertanalyse_1089305#ratings-and-comments.

Rahman, Nayem und Rutz, Dale. 2015. Building Data Warehouses Using Automation. International Journal of Intelligent Information Technologies (IJIIT). 2015, Vol. 11, No. 2, S. 1-22.

Rainardi, Vincent. 2008. Building a Data Warehouse: With Examples in SQL Server. Berkeley, CA: Apress, 2008.

Rajan, Jindal und Abhishek, Acharya. 2019. Federated Data Warehouse Architecture. s.l.: WIPRO, 2019.

Rautenstrauch, Claus und Scholz, André. 1999. Vom Performance-Tuning zum Software Performance Engineering am Beispiel datenbankbasierter Anwendungssysteme, in:. Informatik Spektrum. Vol. 22, No. 4, 1999, S. 261-275.

Schirm, Nicole und Bensberg, Frank. 2018. Wirtschaftlichkeitsbewertung von Cloud-Business-Intelligence & Analytics-Lösungen, in: BI & Analytics in der Cloud: Architektur, Vorgehen und Praxis. [Hrsg.] Ralf Finger. Heidelberg: dpunkt.verlag, 2018. S. 75-90.

Schnider, Dani, et al. 2016. Data Warehouse Blueprints: Business Intelligence in der Praxis. München: Hanser-Verlag, 2016.

Snowflake Computing Logo Clipart in: Pikpng (Stichwort). o. J. [Online] o. J. [Zitat vom: 23. Juli 2021.] https://www.pikpng.com/pngvi/xxJiwm_periscope-data-partners-snowflake-computing-logo-clipart/.

Snowflake Editions, in: Snowflake (Stichwort). o. J. [Online] o. J. [Zitat vom: 13. Juli 2021.] https://docs.snowflake.com/en/user-guide/intro-editions.html.

Snowflake Releases, in: Snowflake (Stichwort). o. J. [Online] o. J. [Zitat vom: 13. Juli 2021.] https://docs.snowflake.com/en/user-guide/intro-releases.html.

Software Engineering Institute. 2011. CMMI® für Entwicklung, Version 1.3. [Online] November 2011. [Zitat vom: 14. August 2021.] https://resources.sei.cmu.edu/asset_files/WhitePaper/2011_019_001_28795.pdf.

Song, Il-Yeol. 2009. Data Warehousing Systems: Foundations and Architectures, in: Encyclopedia of Database Systems. [Hrsg.] M. Tamer Özsu. Boston, MA : Springer, 2009.

Stauffer, Herbert. 2017. Data Warehouse Automation, in: ONLINE-THEMENSPECIAL DATA WAREHOUSE AUTOMATION. [Online] 2017. [Zitat vom: 3. Juni 2021.] http://www.sigs.de/publications/bi/2017/DataWarehouseAutomation/editorial_BIS_OTS_DWA_2017.pdf.

Supported Cloud Platforms, in: Snowflake (Stichwort). o. J. [Online] o. J. [Zitat vom: 13. Juli 2021.] https://docs.snowflake.com/en/user-guide/intro-cloud-platforms.html.

Supported Cloud Regions, in: Snowflake (Stichwort). o. J. [Online] o. J. [Zitat vom: 13. Juli 2021.] https://docs.snowflake.com/en/user-guide/intro-regions.html.

Transaction Processing Performance Council (TPC). 2018. TPC Benchmark™ H Standard Specification Revision 2.18.0. San Francisco: s.n., 2018.

Use Docker Compose, in: Docker (Stichwort). o. J. [Online] o. J. [Zitat vom: 10. Juli 2021.] https://docs.docker.com/get-started/08_using_compose/.

Virtuelle Linux-Computer in Azure, in: Microsoft (Stichwort). 2019. [Online] 11. November 2019. [Zitat vom: 12. Juli 2021.] https://docs.microsoft.com/de-de/azure/virtual-machines/linux/overview.

Visuelle Datawarehouse Entwicklung - effizienter geht nicht, in: Ventum-consulting (Stichwort). o. J. [Online] o. J. [Zitat vom: 12. Juli 2021.] https://www.ventum-consulting.com/de/portfolio/business-intelligence/datavault-builder.html.

Vivenzio, Alberto und Vivenzio, Domenico. 2013. Testmanagement bei SAP-Projekten: erfolgreich planen - steuern - reporten bei der Einführung von SAP-Banking. Wiesbaden: Springer Vieweg, 2013.

Was ist Azure Files?, in: Microsoft (Stichwort). 2020. [Online] 15. September 2020. [Zitat vom: 12. Juli 2021.] https://docs.microsoft.com/de-de/azure/storage/files/storage-files-introduction.

Was ist Azure SQL-Datenbank?, in: Microsoft (Stichwort). 2020. [Online] 21. September 2020. [Zitat vom: 13. Juli 2021.] https://docs.microsoft.com/de-de/azure/azure-sql/database/sql-database-paas-overview.

Wells, Dave. 2014. Relieving the Pain of the BI Back Room with Data Warehouse Automation, in: Tdwi. [Online] 15. April 2014. [Zitat vom: 3. Juni 2021.] https://tdwi.org/articles/2014/04/15/data-warehouse-automation.aspx.

Zangemeister, Christof. 2014. Nutzwertanalyse in der Systemtechnik: Eine Methode zur multidimensionalen Bewertung und Auswahl von Projektalternativen. 5., erweiterte Auflage. Winnemark: Selbstverlag, 2014.

Zeiler, Gregor. 2017. Data Warehouse Automation führt Unternehmen in die Zukunft Automatisierung ist der Schlüssel, in: ONLINE-THEMENSPECIAL DATA WAREHOUSE AUTOMATION. [Online] 2017. [Zitat vom: 3. Juni 2021.] http://www.sigs.de/publications/bi/2017/DataWarehouseAutomation/zeiler_BIS_OTS_DWA_2017.pdf.

Eidesstattliche Versicherung:

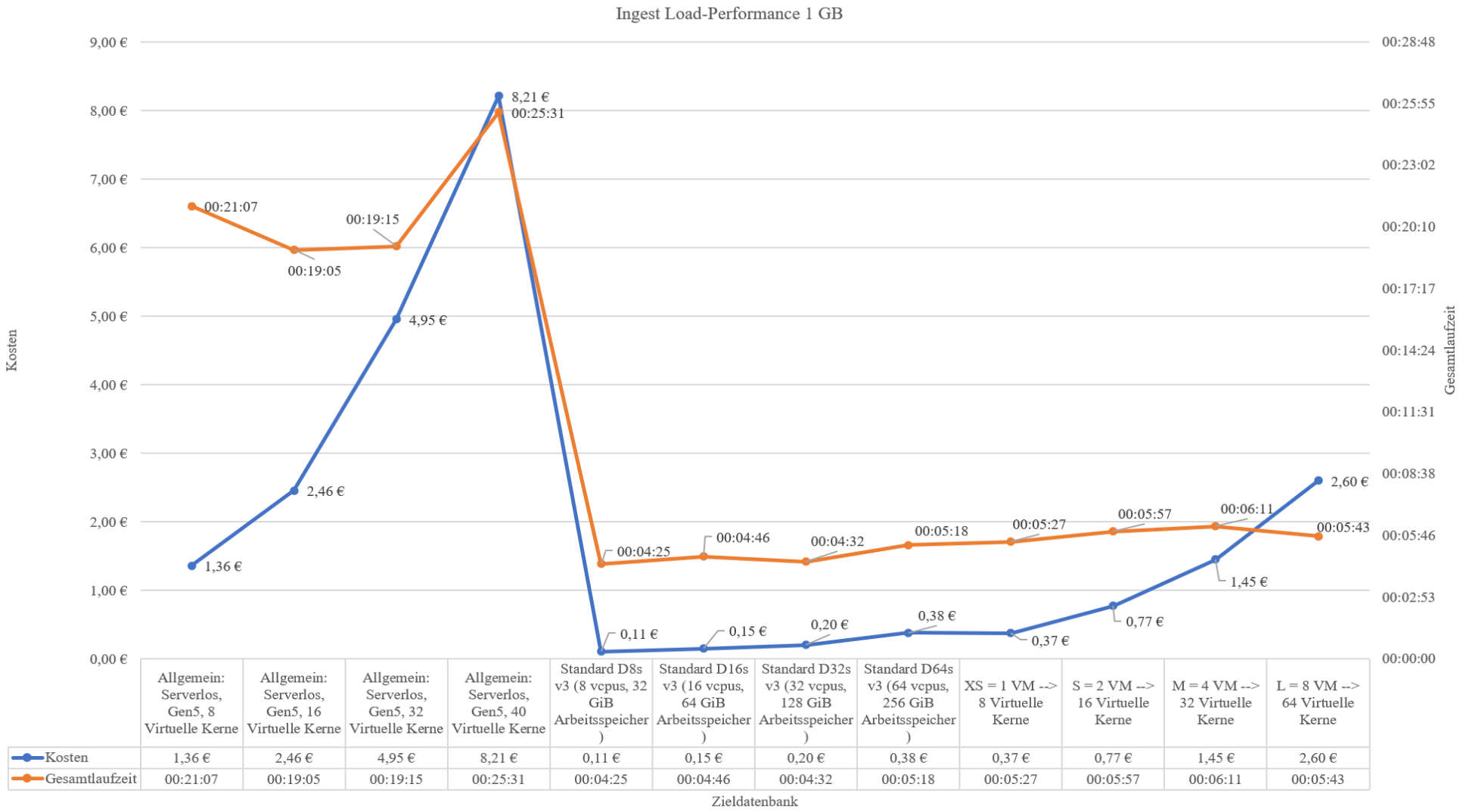
Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

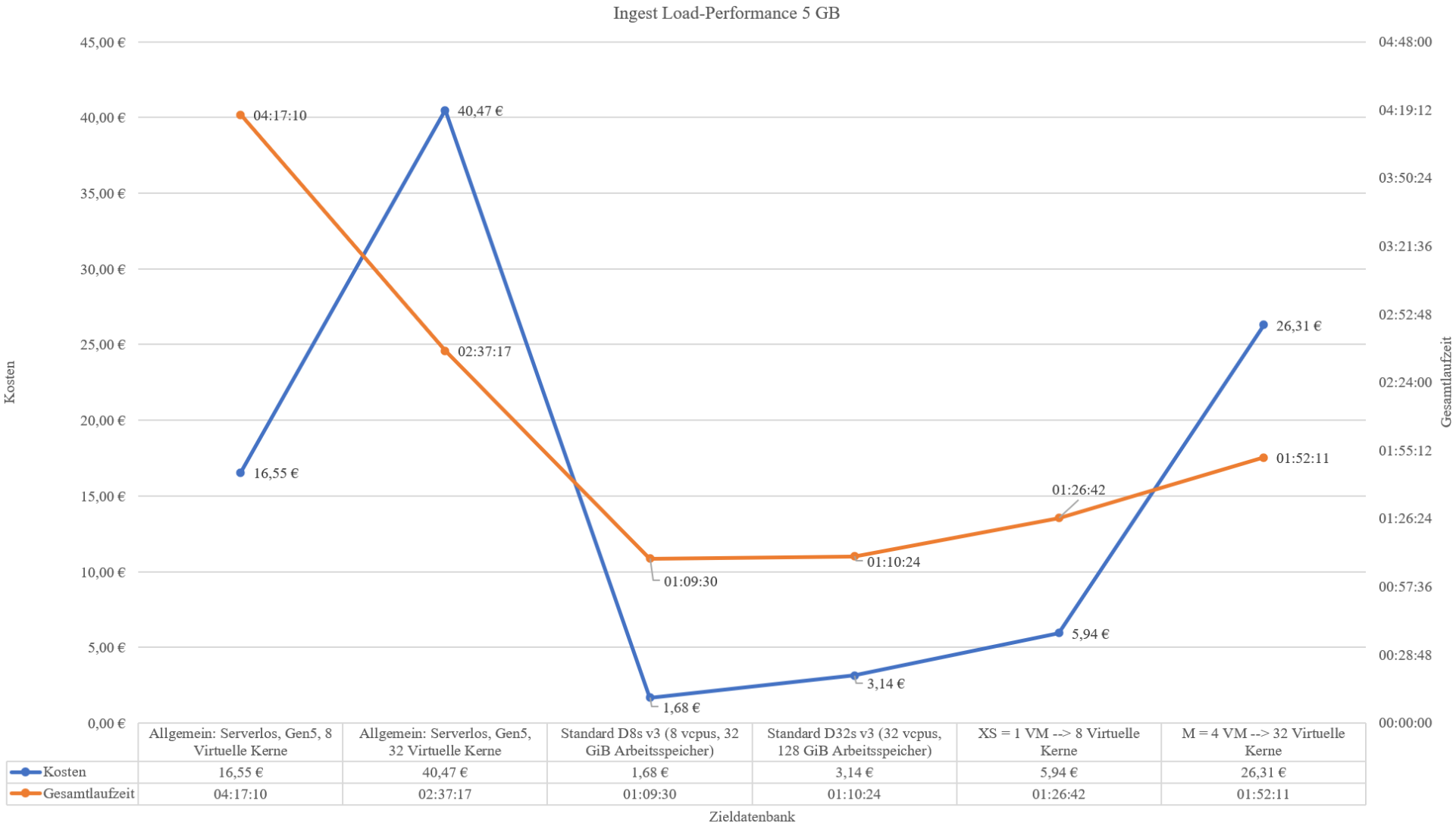
Rosbach v. d. H. 20.10.2021 _____

Ort, Datum Unterschrift

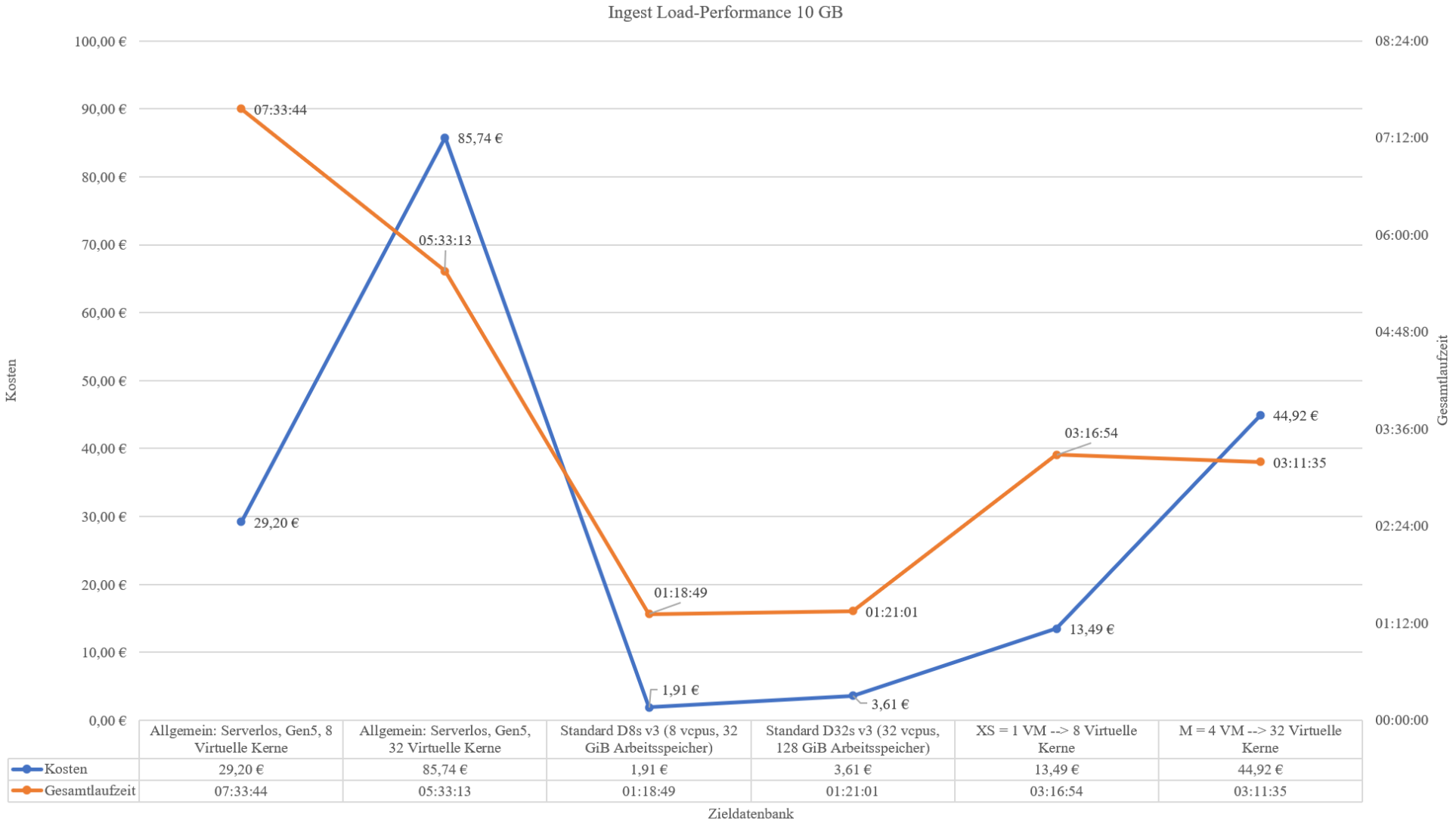
Anhang 1



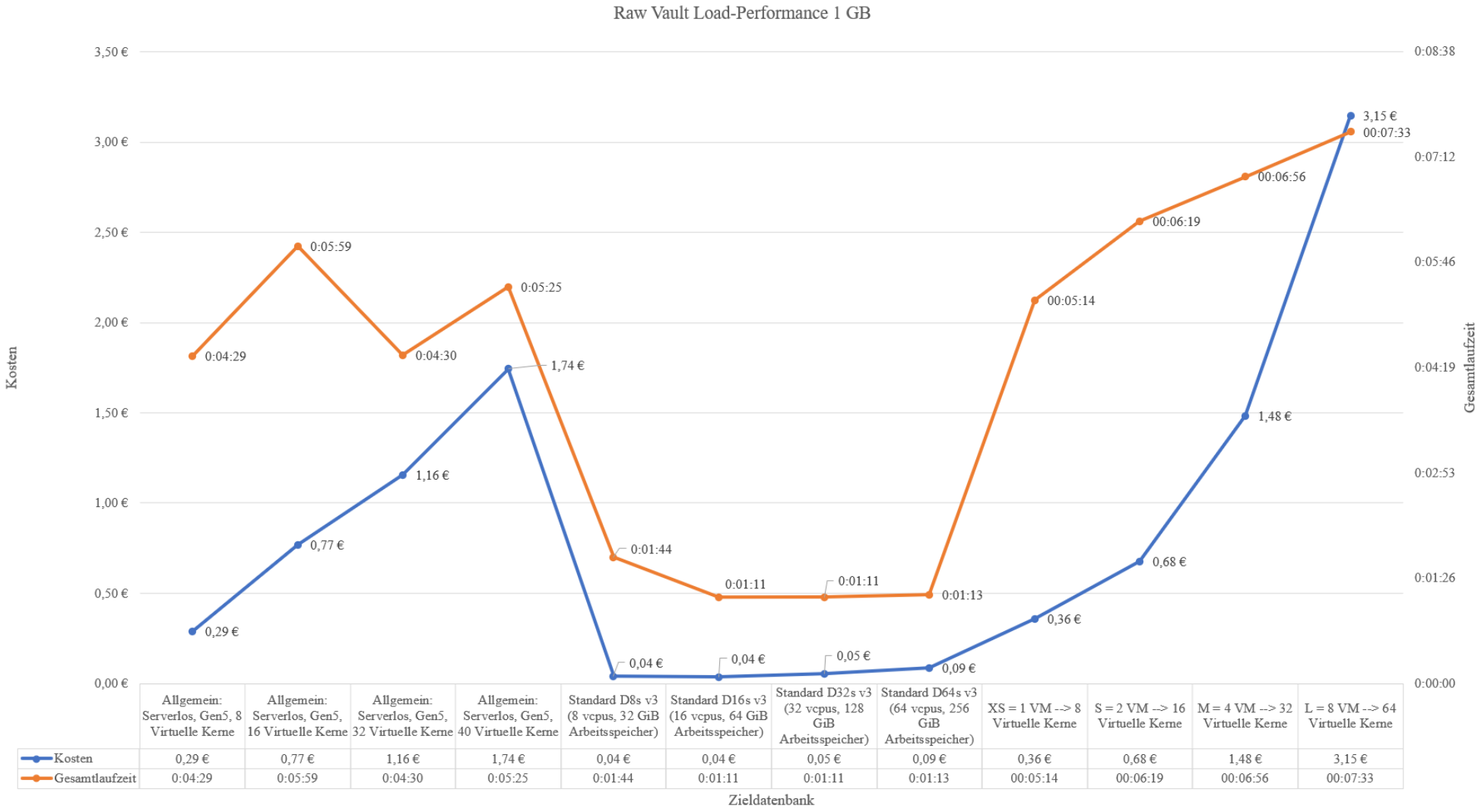
	Allgemein: Serverlos, Gen5, 8 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 16 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 32 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 40 Virtuelle Kerne	Standard D8s v3 (8 vcpus, 32 GiB Arbeitsspeicher)	Standard D16s v3 (16 vcpus, 64 GiB Arbeitsspeicher)	Standard D32s v3 (32 vcpus, 128 GiB Arbeitsspeicher)	Standard D64s v3 (64 vcpus, 256 GiB Arbeitsspeicher)	XS = 1 VM --> 8 Virtuelle Kerne	S = 2 VM --> 16 Virtuelle Kerne	M = 4 VM --> 32 Virtuelle Kerne	L = 8 VM --> 64 Virtuelle Kerne
Kosten	1,36 €	2,46 €	4,95 €	8,21 €	0,11 €	0,15 €	0,20 €	0,38 €	0,37 €	0,77 €	1,45 €	2,60 €
Gesamtlaufzeit	00:21:07	00:19:05	00:19:15	00:25:31	00:04:25	00:04:46	00:04:32	00:05:18	00:05:27	00:05:57	00:06:11	00:05:43



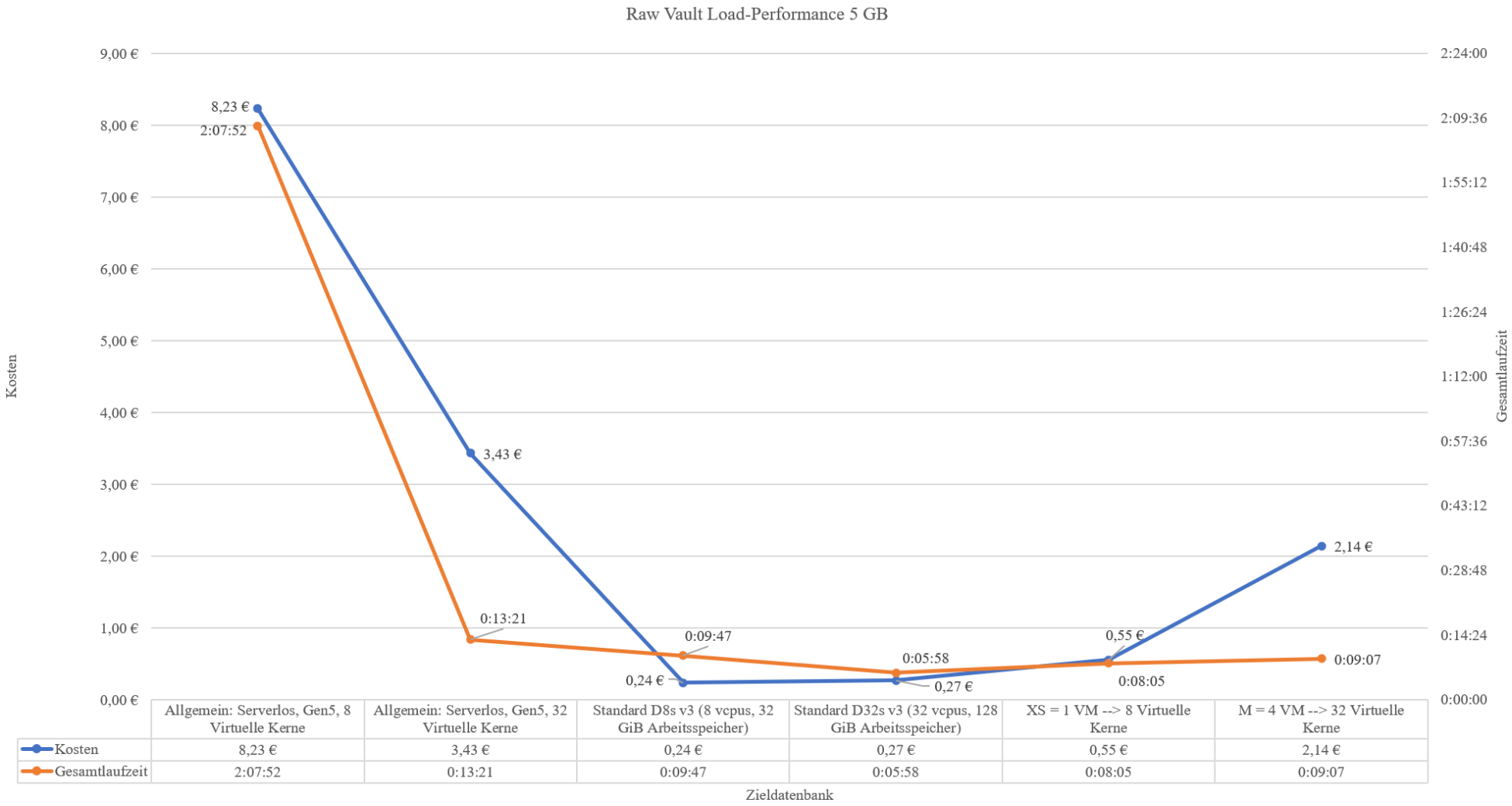
	Allgemein: Serverlos, Gen5, 8 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 32 Virtuelle Kerne	Standard D8s v3 (8 vcpus, 32 GiB Arbeitsspeicher)	Standard D32s v3 (32 vcpus, 128 GiB Arbeitsspeicher)	XS = 1 VM --> 8 Virtuelle Kerne	M = 4 VM --> 32 Virtuelle Kerne
Kosten	16,55 €	40,47 €	1,68 €	3,14 €	5,94 €	26,31 €
Gesamtlaufzeit	04:17:10	02:37:17	01:09:30	01:10:24	01:26:42	01:52:11



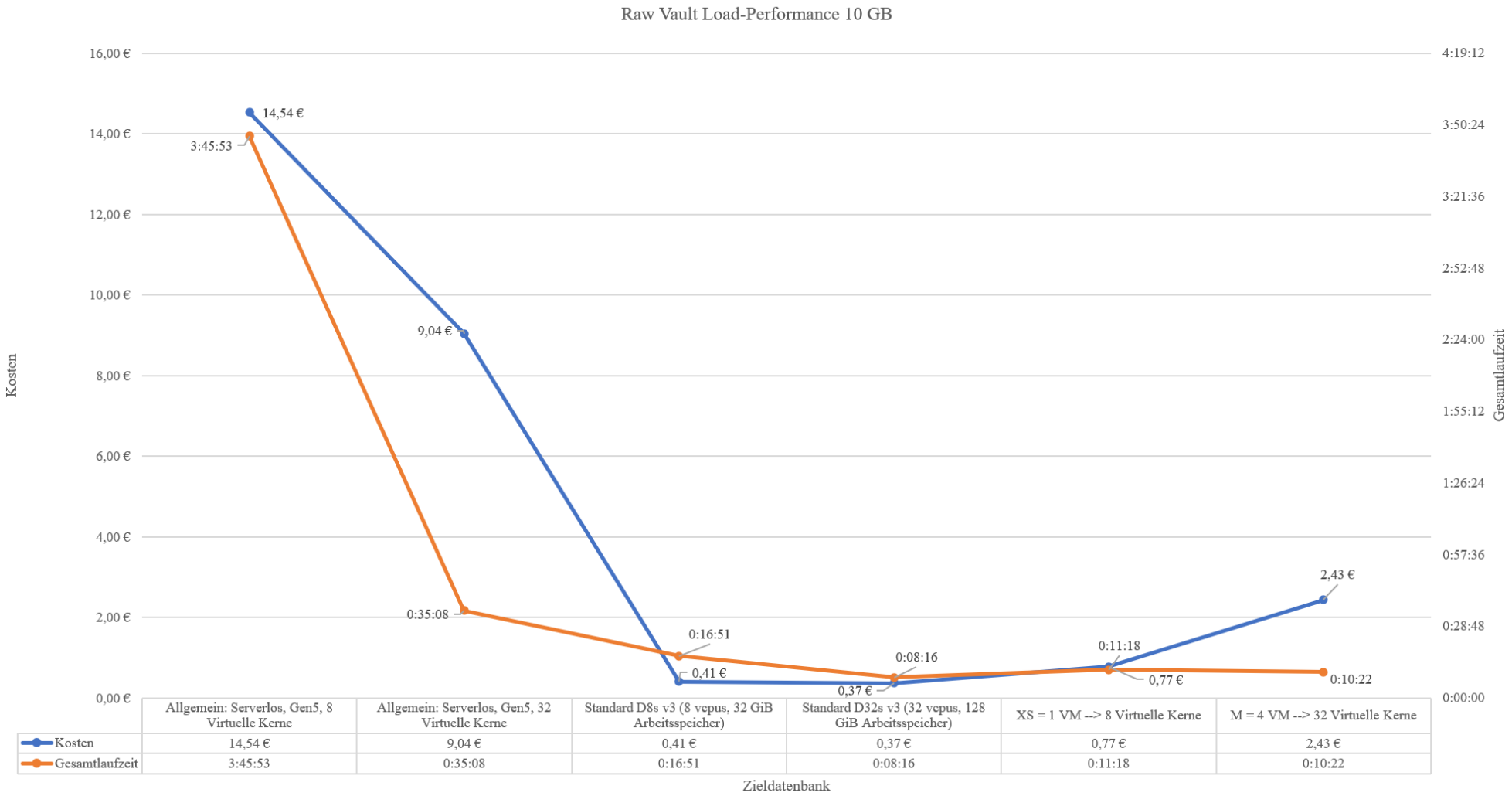
	Allgemein: Serverlos, Gen5, 8 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 32 Virtuelle Kerne	Standard D8s v3 (8 vcpus, 32 GiB Arbeitsspeicher)	Standard D32s v3 (32 vcpus, 128 GiB Arbeitsspeicher)	XS = 1 VM --> 8 Virtuelle Kerne	M = 4 VM --> 32 Virtuelle Kerne
Kosten	29,20 €	85,74 €	1,91 €	3,61 €	13,49 €	44,92 €
Gesamtlaufzeit	07:33:44	05:33:13	01:18:49	01:21:01	03:16:54	03:11:35

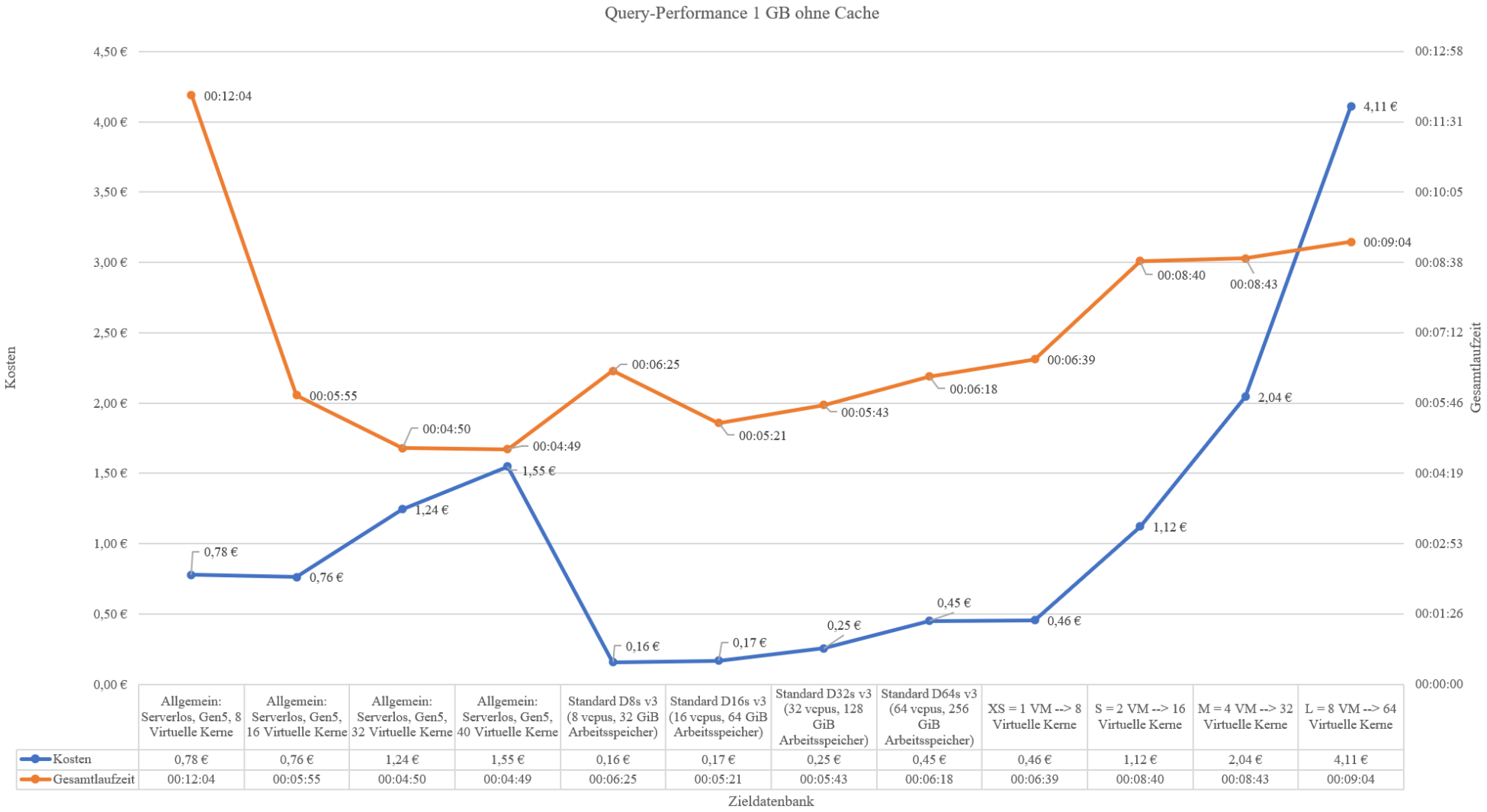


	Allgemein: Serverlos, Gen5, 8 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 16 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 32 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 40 Virtuelle Kerne	Standard D8s v3 (8 vcpus, 32 GiB Arbeitsspeicher)	Standard D16s v3 (16 vcpus, 64 GiB Arbeitsspeicher)	Standard D32s v3 (32 vcpus, 128 GiB Arbeitsspeicher)	Standard D64s v3 (64 vcpus, 256 GiB Arbeitsspeicher)	XS = 1 VM --> 8 Virtuelle Kerne	S = 2 VM --> 16 Virtuelle Kerne	M = 4 VM --> 32 Virtuelle Kerne	L = 8 VM --> 64 Virtuelle Kerne
Kosten	0,29 €	0,77 €	1,16 €	1,74 €	0,04 €	0,04 €	0,05 €	0,09 €	0,36 €	0,68 €	1,48 €	3,15 €
Gesamtlaufzeit	0:04:29	0:05:59	0:04:30	0:05:25	0:01:44	0:01:11	0:01:11	0:01:13	00:05:14	00:06:19	00:06:56	00:07:33



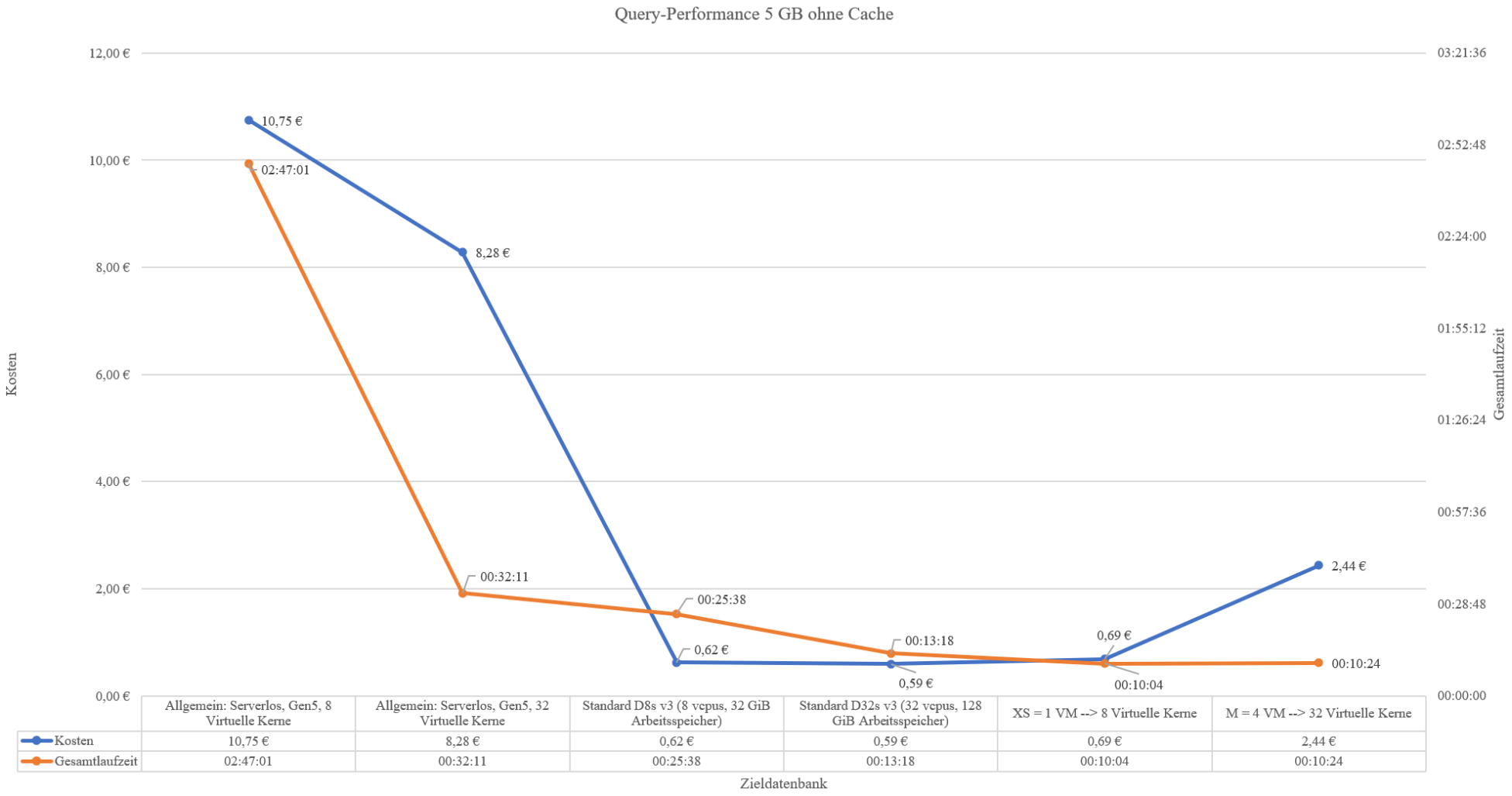
	Allgemein: Serverlos, Gen5, 8 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 32 Virtuelle Kerne	Standard D8s v3 (8 vcpus, 32 GiB Arbeitsspeicher)	Standard D32s v3 (32 vcpus, 128 GiB Arbeitsspeicher)	XS = 1 VM --> 8 Virtuelle Kerne	M = 4 VM --> 32 Virtuelle Kerne
Kosten	8,23 €	3,43 €	0,24 €	0,27 €	0,55 €	2,14 €
Gesamtlaufzeit	2:07:52	0:13:21	0:09:47	0:05:58	0:08:05	0:09:07

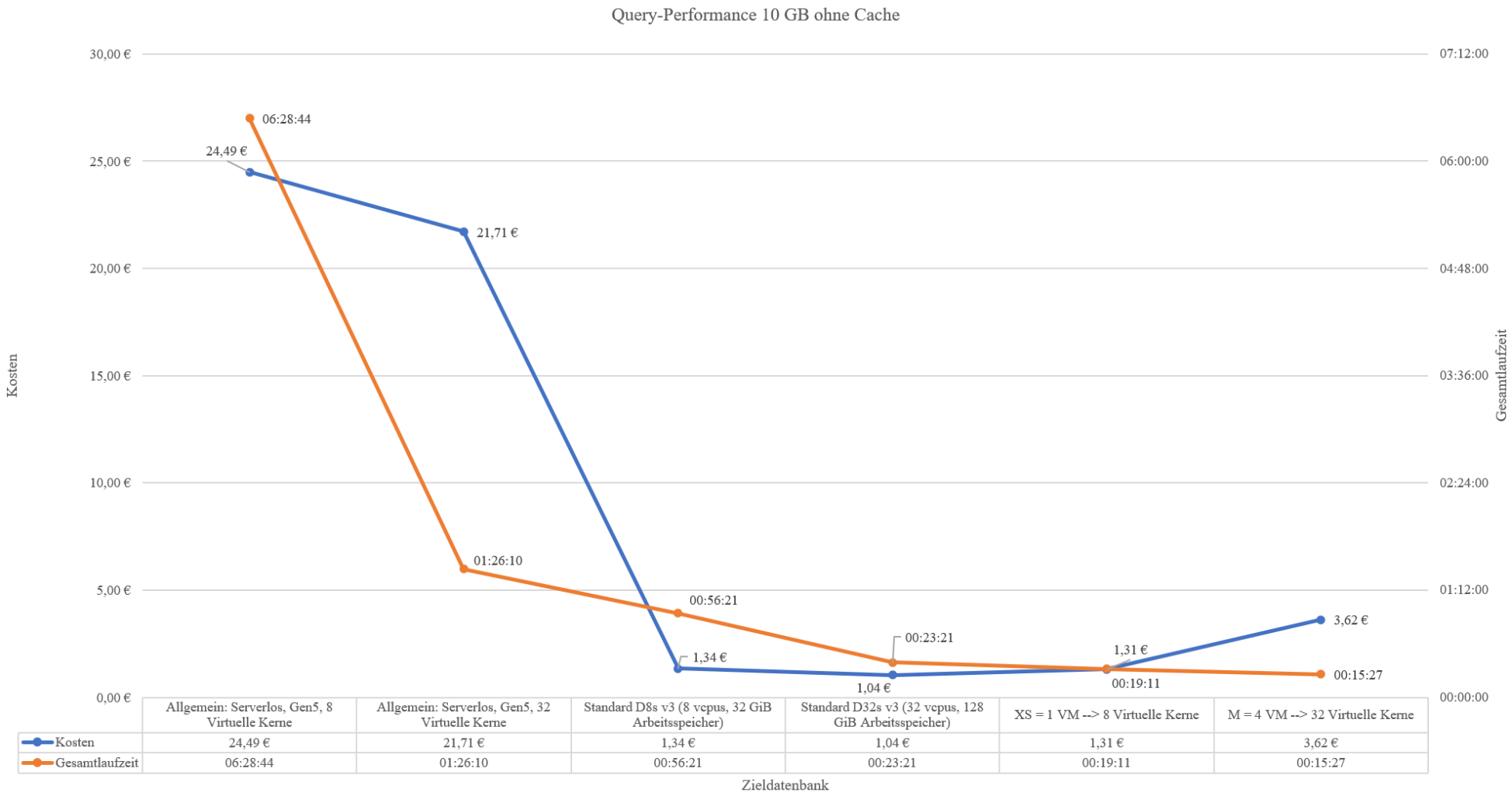

Raw Vault Load-Performance 10 GB
Kosten
Gesamtlaufzeit
16,00 €
14,00 €
12,00 €
10,00 €
8,00 €
6,00 €
4,00 €
2,00 €
0,00 €
4:19:12
3:50:24
3:21:36
2:52:48
2:24:00
1:55:12
1:26:24
0:57:36
0:28:48
0:00:00
14,54 €
3:45:53
9,04 €
0:35:08
0,41 €
0:16:51
0,37 €
0:08:16
0,77 €
0:11:18
2,43 €
0:10:22
Allgemein: Serverlos, Gen5, 8 Virtuelle Kerne
Allgemein: Serverlos, Gen5, 32 Virtuelle Kerne
Standard D8s v3 (8 vcpus, 32 GiB Arbeitsspeicher)
Standard D32s v3 (32 vcpus, 128 GiB Arbeitsspeicher)
XS = 1 VM --> 8 Virtuelle Kerne
M = 4 VM --> 32 Virtuelle Kerne
Kosten
14,54 €
9,04 €
0,41 €
0,37 €
0,77 €
2,43 €
Gesamtlaufzeit
3:45:53
0:35:08
0:16:51
0:08:16
0:11:18
0:10:22
Zieldatenbank

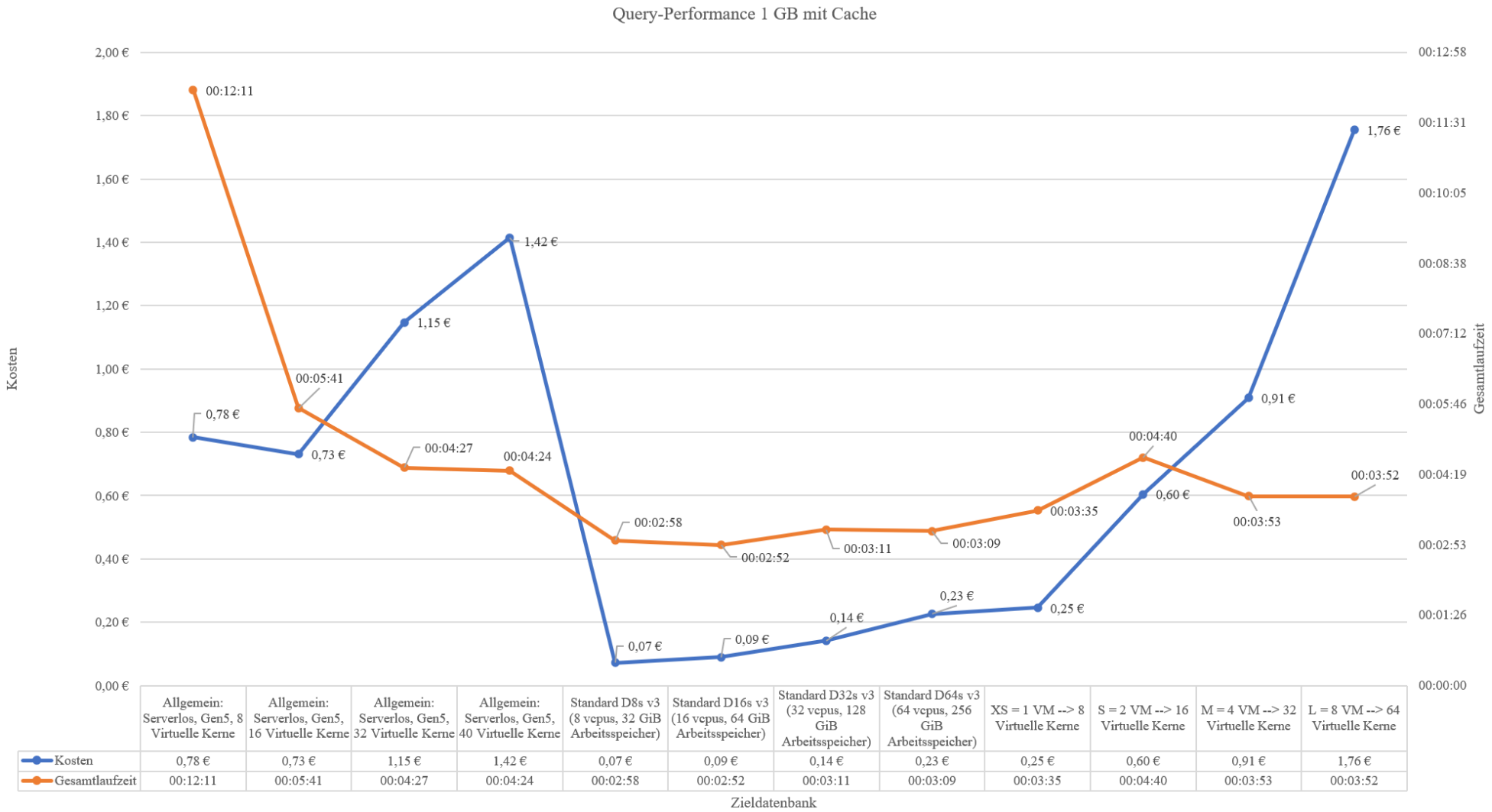


	Allgemein: Serverlos, Gen5, 8 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 16 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 32 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 40 Virtuelle Kerne	Standard D8s v3 (8 vcpus, 32 GiB Arbeitsspeicher)	Standard D16s v3 (16 vcpus, 64 GiB Arbeitsspeicher)	Standard D32s v3 (32 vcpus, 128 GiB Arbeitsspeicher)	Standard D64s v3 (64 vcpus, 256 GiB Arbeitsspeicher)	XS = 1 VM --> 8 Virtuelle Kerne	S = 2 VM --> 16 Virtuelle Kerne	M = 4 VM --> 32 Virtuelle Kerne	L = 8 VM --> 64 Virtuelle Kerne
Kosten	0,78 €	0,76 €	1,24 €	1,55 €	0,16 €	0,17 €	0,25 €	0,45 €	0,46 €	1,12 €	2,04 €	4,11 €
Gesamtlaufzeit	00:12:04	00:05:55	00:04:50	00:04:49	00:06:25	00:05:21	00:05:43	00:06:18	00:06:39	00:08:40	00:08:43	00:09:04

Zieldatenbank

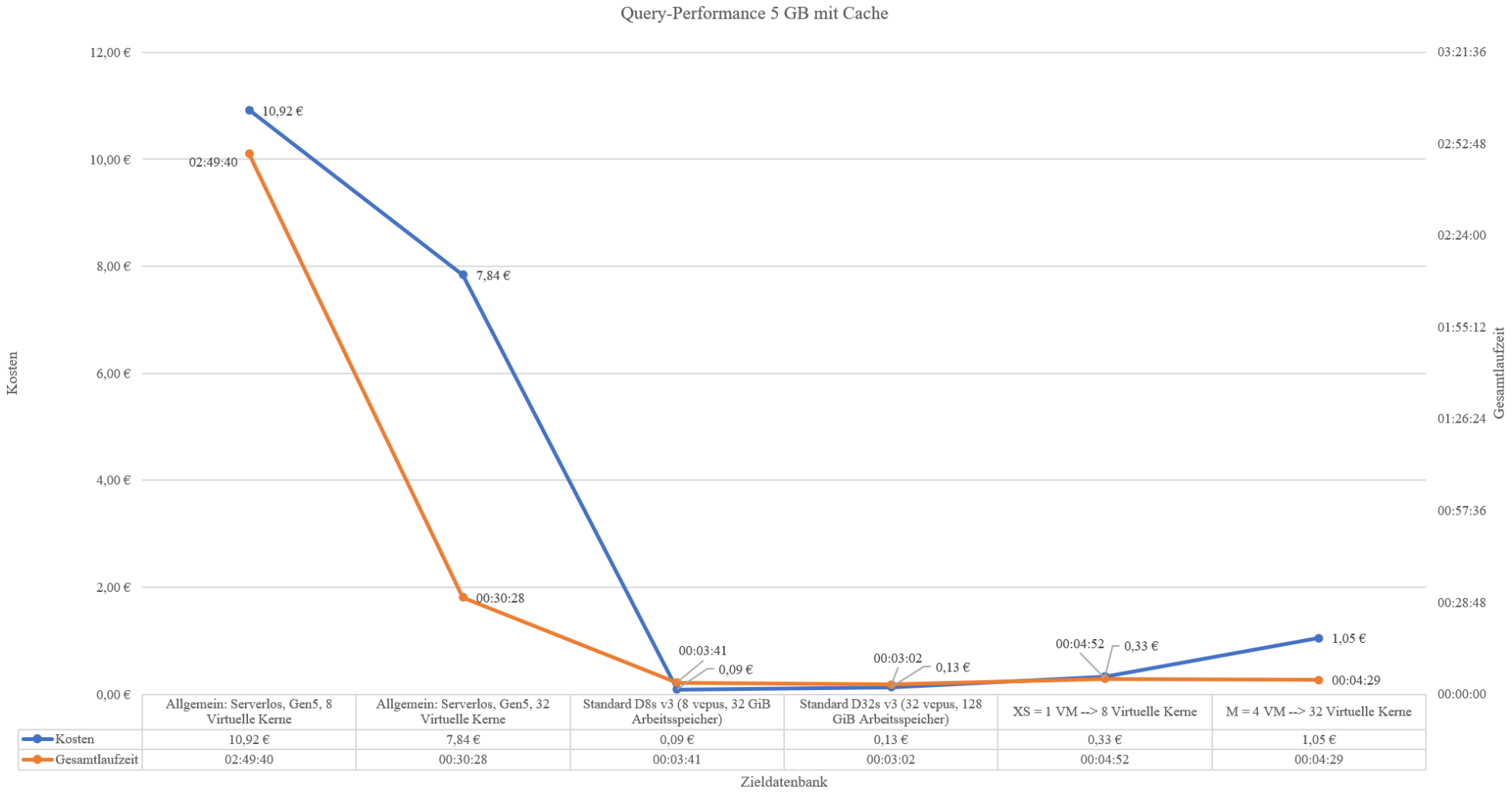

Query-Performance 5 GB ohne Cache
12,00 €
10,00 €
8,00 €
6,00 €
4,00 €
2,00 €
0,00 €
Kosten
03:21:36
02:52:48
02:24:00
01:55:12
01:26:24
00:57:36
00:28:48
00:00:00
Gesamtlaufzeit
10,75 €
02:47:01
8,28 €
00:32:11
00:25:38
0,62 €
00:13:18
0,59 €
0,69 €
00:10:04
2,44 €
00:10:24
Allgemein: Serverlos, Gen5, 8 Virtuelle Kerne
Allgemein: Serverlos, Gen5, 32 Virtuelle Kerne
Standard D8s v3 (8 vcpus, 32 GiB Arbeitsspeicher)
Standard D32s v3 (32 vcpus, 128 GiB Arbeitsspeicher)
XS = 1 VM --> 8 Virtuelle Kerne
M = 4 VM --> 32 Virtuelle Kerne
Kosten
10,75 €
8,28 €
0,62 €
0,59 €
0,69 €
2,44 €
Gesamtlaufzeit
02:47:01
00:32:11
00:25:38
00:13:18
00:10:04
00:10:24
Zieldatenbank

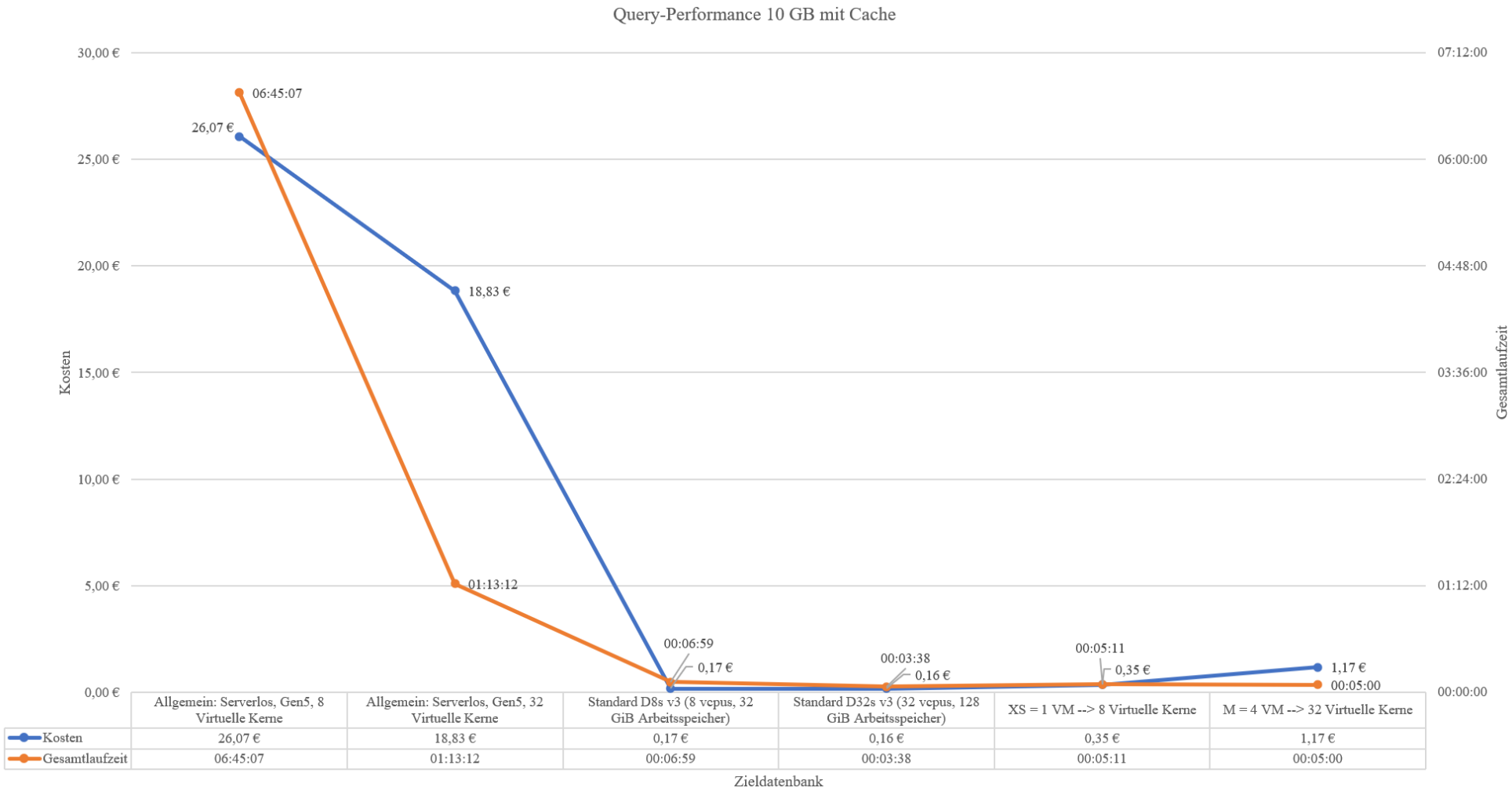

Query-Performance 10 GB ohne Cache
30,00 €
25,00 €
20,00 €
15,00 €
10,00 €
5,00 €
0,00 €
07:12:00
06:00:00
04:48:00
03:36:00
02:24:00
01:12:00
00:00:00
Kosten
Gesamtlaufzeit
24,49 €
06:28:44
21,71 €
01:26:10
1,34 €
00:56:21
1,04 €
00:23:21
1,31 €
00:19:11
3,62 €
00:15:27
Allgemein: Serverlos, Gen5, 8 Virtuelle Kerne
Allgemein: Serverlos, Gen5, 32 Virtuelle Kerne
Standard D8s v3 (8 vcpus, 32 GiB Arbeitsspeicher)
Standard D32s v3 (32 vcpus, 128 GiB Arbeitsspeicher)
XS = 1 VM --> 8 Virtuelle Kerne
M = 4 VM --> 32 Virtuelle Kerne
Kosten
24,49 €
21,71 €
1,34 €
1,04 €
1,31 €
3,62 €
Gesamtlaufzeit
06:28:44
01:26:10
00:56:21
00:23:21
00:19:11
00:15:27
Zieldatenbank



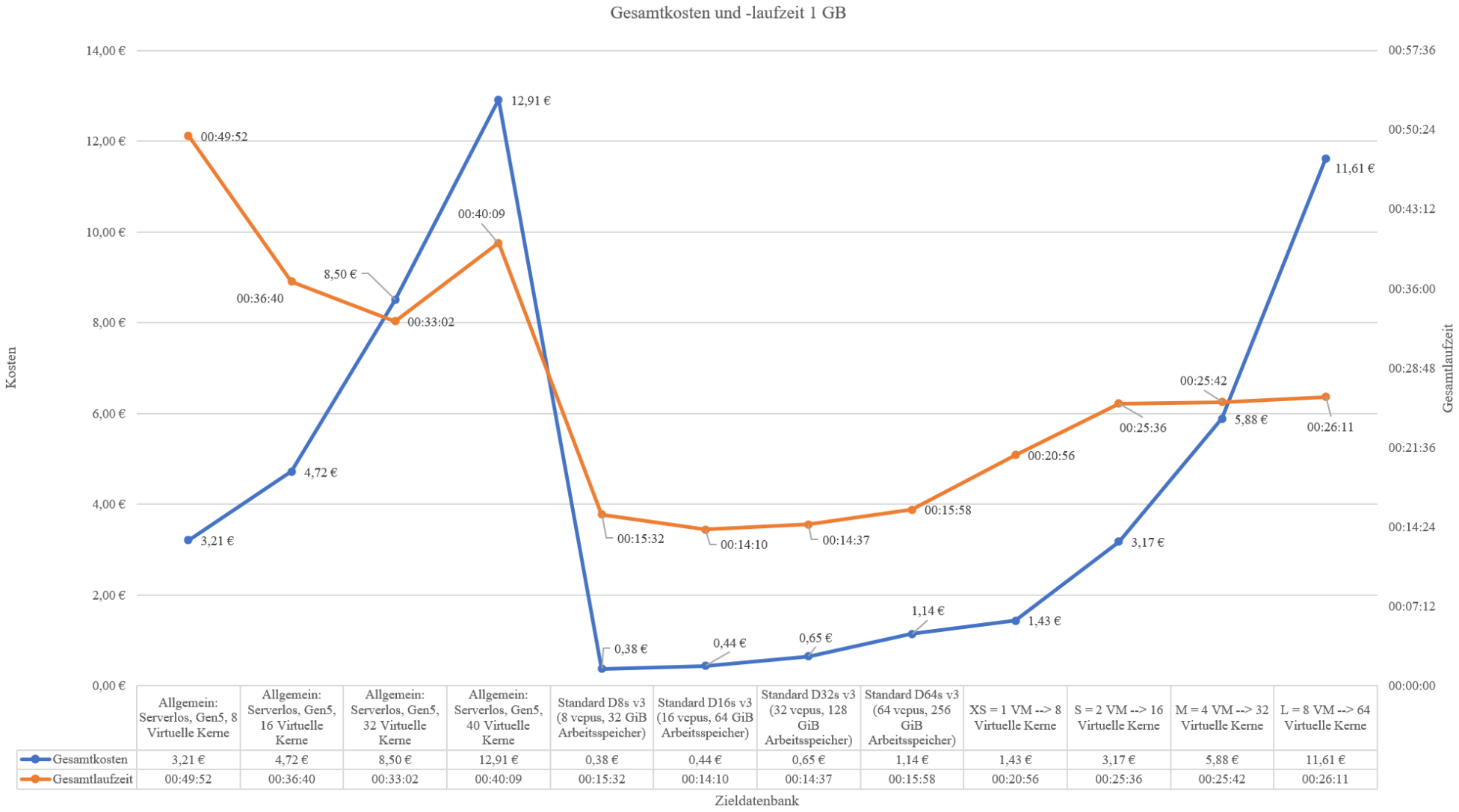
	Allgemein: Serverlos, Gen5, 8 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 16 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 32 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 40 Virtuelle Kerne	Standard D8s v3 (8 vcpus, 32 GiB Arbeitsspeicher)	Standard D16s v3 (16 vcpus, 64 GiB Arbeitsspeicher)	Standard D32s v3 (32 vcpus, 128 GiB Arbeitsspeicher)	Standard D64s v3 (64 vcpus, 256 GiB Arbeitsspeicher)	XS = 1 VM --> 8 Virtuelle Kerne	S = 2 VM --> 16 Virtuelle Kerne	M = 4 VM --> 32 Virtuelle Kerne	L = 8 VM --> 64 Virtuelle Kerne
Kosten	0,78 €	0,73 €	1,15 €	1,42 €	0,07 €	0,09 €	0,14 €	0,23 €	0,25 €	0,60 €	0,91 €	1,76 €
Gesamtlaufzeit	00:12:11	00:05:41	00:04:27	00:04:24	00:02:58	00:02:52	00:03:11	00:03:09	00:03:35	00:04:40	00:03:53	00:03:52

Zieldatenbank


Query-Performance 5 GB mit Cache
12,00 €
10,00 €
8,00 €
6,00 €
4,00 €
2,00 €
0,00 €
Kosten
03:21:36
02:52:48
02:24:00
01:55:12
01:26:24
00:57:36
00:28:48
00:00:00
Gesamtlaufzeit
10,92 €
02:49:40
7,84 €
00:30:28
00:03:41
0,09 €
00:03:02
0,13 €
00:04:52
0,33 €
1,05 €
00:04:29
Allgemein: Serverlos, Gen5, 8 Virtuelle Kerne
Allgemein: Serverlos, Gen5, 32 Virtuelle Kerne
Standard D8s v3 (8 vcpus, 32 GiB Arbeitsspeicher)
Standard D32s v3 (32 vcpus, 128 GiB Arbeitsspeicher)
XS = 1 VM --> 8 Virtuelle Kerne
M = 4 VM --> 32 Virtuelle Kerne
Kosten
10,92 €
7,84 €
0,09 €
0,13 €
0,33 €
1,05 €
Gesamtlaufzeit
02:49:40
00:30:28
00:03:41
00:03:02
00:04:52
00:04:29
Zieldatenbank

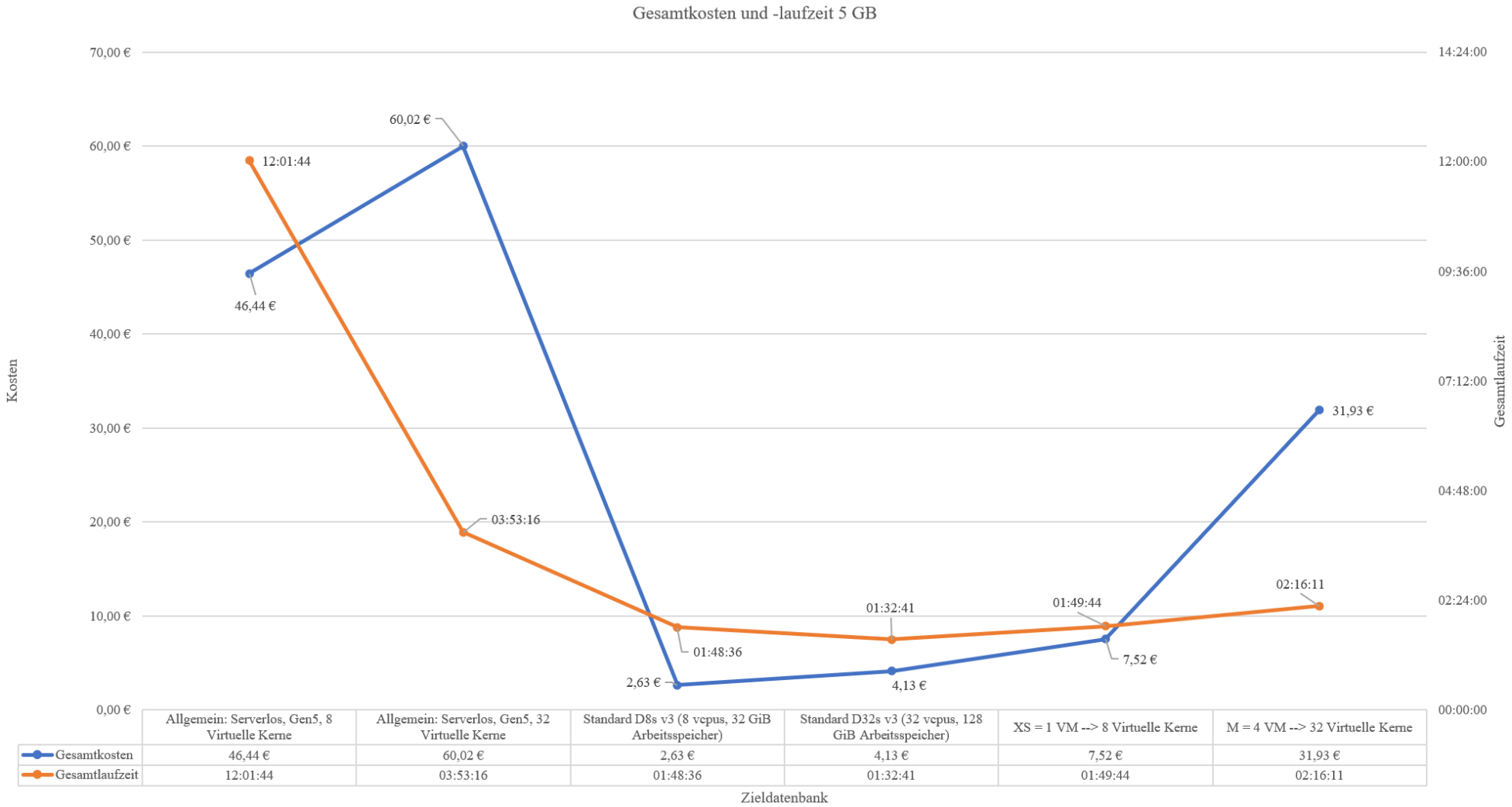


	Allgemein: Serverlos, Gen5, 8 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 32 Virtuelle Kerne	Standard D8s v3 (8 vcpus, 32 GiB Arbeitsspeicher)	Standard D32s v3 (32 vcpus, 128 GiB Arbeitsspeicher)	XS = 1 VM --> 8 Virtuelle Kerne	M = 4 VM --> 32 Virtuelle Kerne
Kosten	26,07 €	18,83 €	0,17 €	0,16 €	0,35 €	1,17 €
Gesamtlaufzeit	06:45:07	01:13:12	00:06:59	00:03:38	00:05:11	00:05:00

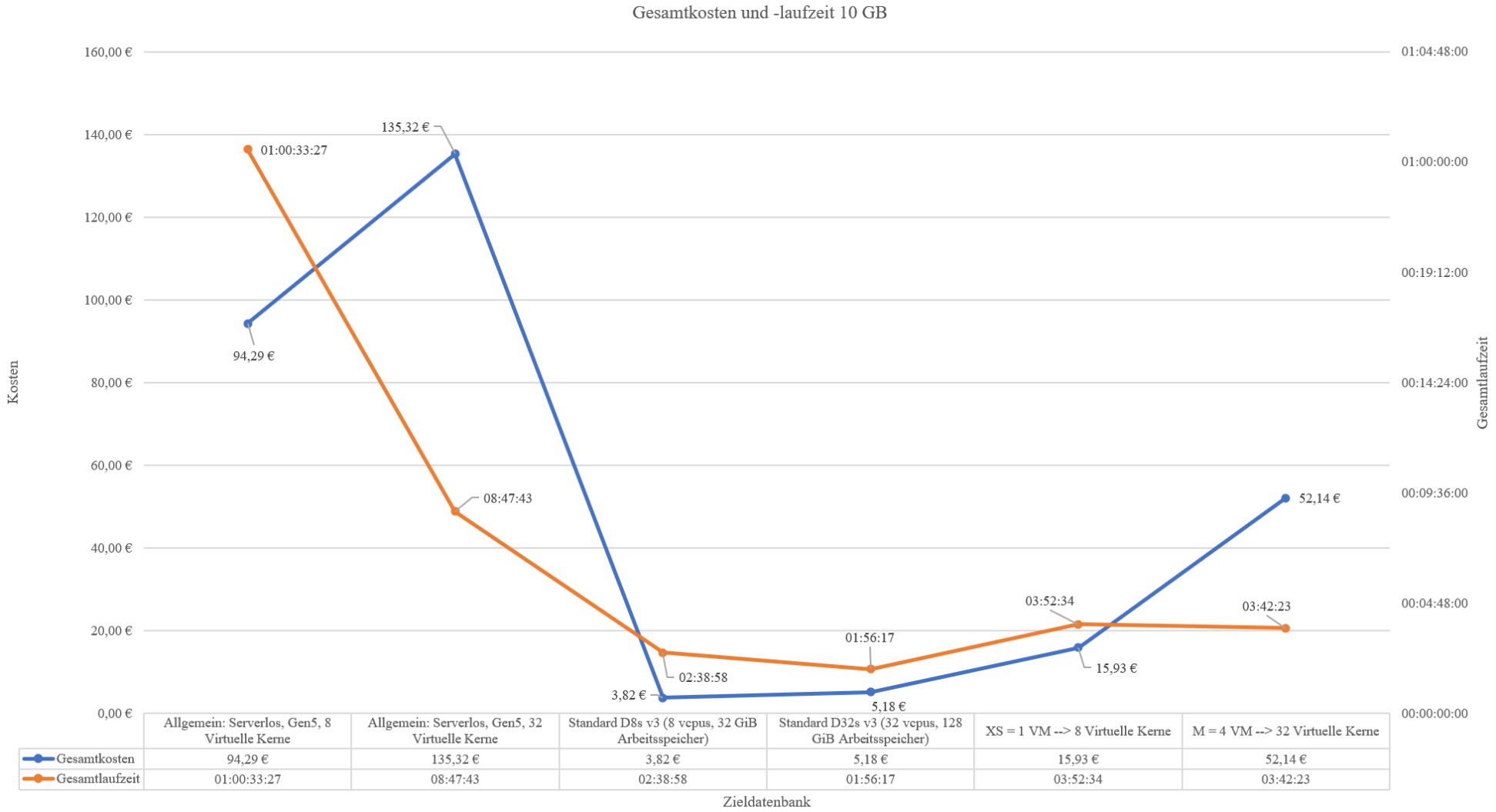


	Allgemein: Serverlos, Gen5, 8 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 16 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 32 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 40 Virtuelle Kerne	Standard D8s v3 (8 vcpus, 32 GiB Arbeitsspeicher)	Standard D16s v3 (16 vcpus, 64 GiB Arbeitsspeicher)	Standard D32s v3 (32 vcpus, 128 GiB Arbeitsspeicher)	Standard D64s v3 (64 vcpus, 256 GiB Arbeitsspeicher)	XS = 1 VM --> 8 Virtuelle Kerne	S = 2 VM --> 16 Virtuelle Kerne	M = 4 VM --> 32 Virtuelle Kerne	L = 8 VM --> 64 Virtuelle Kerne
Gesamtkosten	3,21 €	4,72 €	8,50 €	12,91 €	0,38 €	0,44 €	0,65 €	1,14 €	1,43 €	3,17 €	5,88 €	11,61 €
Gesamtlaufzeit	00:49:52	00:36:40	00:33:02	00:40:09	00:15:32	00:14:10	00:14:37	00:15:58	00:20:56	00:25:36	00:25:42	00:26:11

Zieldatenbank



	Allgemein: Serverlos, Gen5, 8 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 32 Virtuelle Kerne	Standard D8s v3 (8 vcpus, 32 GiB Arbeitsspeicher)	Standard D32s v3 (32 vcpus, 128 GiB Arbeitsspeicher)	XS = 1 VM --> 8 Virtuelle Kerne	M = 4 VM --> 32 Virtuelle Kerne
Gesamtkosten	46,44 €	60,02 €	2,63 €	4,13 €	7,52 €	31,93 €
Gesamtlaufzeit	12:01:44	03:53:16	01:48:36	01:32:41	01:49:44	02:16:11



	Allgemein: Serverlos, Gen5, 8 Virtuelle Kerne	Allgemein: Serverlos, Gen5, 32 Virtuelle Kerne	Standard D8s v3 (8 vcpus, 32 GiB Arbeitsspeicher)	Standard D32s v3 (32 vcpus, 128 GiB Arbeitsspeicher)	XS = 1 VM --> 8 Virtuelle Kerne	M = 4 VM --> 32 Virtuelle Kerne
Gesamtkosten	94,29 €	135,32 €	3,82 €	5,18 €	15,93 €	52,14 €
Gesamtlaufzeit	01:00:33:27	08:47:43	02:38:58	01:56:17	03:52:34	03:42:23