Institut für Architektur von Anwendungssystemen (IAAS)
Universität Stuttgart
Universitätsstraße 38
70569 Stuttgart

Diplomarbeit Nr. 3360

Entwicklung einer Methodik für die Migration der Datenbankschicht in die Cloud

Thomas Bachmann

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Frank Leymann
Betreuer:	Steve Strauch
begonnen am:	10. Juli 2012
beendet am:	5. September 2012
CR-Klassifikation:	C.2.4, D.2.7, D.2.11, D.2.12, E.0, H.2.1, H.2.4, H.2.7, H.3.4, H.3.5, H.4.2

Abstract

This diploma thesis identifies different Cloud data migration scenarios like outsourcing, replication, data synchronization, backup, archiving, scalability enhancement, Cloud burst, sharding or data import and puts them into a generic Cloud data migration methodology that includes all mentioned scenarios. Next to relational database management systems this methodology also supports NoSQL data stores, BLOB stores and content delivery networks. Using the generic methodology a Cloud data migration tool was developed which supports the decision process of selecting a Cloud data store and helps refactoring the application architecture by suggesting Cloud data patterns and potential adaptions of the network, data access and application layer. Finally the migration tool is able to copy data from a source system like MySQL server to a target system like Amazon RDS (MySQL, SQL Server, and Oracle), Azure SQL Database and Amazon SimpleDB.

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Abkürzungsverzeichnis

Abkürzung	Bedeutung
ACID	Atomicity, Consistency, Isolation, Durability
AWS	Amazon Web Services
BLOB	Binary Large Object
CDN	Content Delivery Network
DaaS	Database-as-a-Service
DAL	Data Access Layer
DBA	Datenbankadministrator
DBL	Database Layer
DBMS	Database Managementsystem
DSS	Decision Support System
ETL	Extract-Transform-Load
IaaS	Infrastructure-as-a-Service
NoSQL	Not only SQL oder „Kein" SQL
PaaS	Platform-as-a-Service
PCI DSS	Payment Card Industry Data Security Standard
RDBMS	Relationales Datenbank Managementsystem
RDS	Relational Database Service
SOA	Service-Oriented Architecture
SQL	Structured Query Language
TSQL	Transact-SQL

Inhaltsverzeichnis

1 Einführung

Im Einführungskapitel wird das Thema dieser Arbeit motiviert und der Rahmen der Arbeit definiert. Dabei wird eine Abgrenzung zu tangierenden Themengebieten vorgenommen und die Struktur der Arbeit erläutert.

1.1 Motivation

Cloud Computing wird immer populärer und bringt viele Vorteile mit sich im Vergleich zum Betrieb von Anwendungen im eigenen Rechenzentrum. So können aus Sicht von Cloud Computing Konsumenten z.B. einmalig hohe Investitionsaufwände in regelmäßige kleinere Betriebsausgaben umgewandelt (pay-per-use), insgesamt Kosten gespart, die Flexibilität bei variabler Auslastung verbessert (Elastizität) und eine bessere Skalierbarkeit sowie höhere Verfügbarkeit erreicht werden. Aus Cloud Anbietersicht kann außerdem z.B. durch Mandantenfähigkeit [1], [2] eine höhere Ressourcenauslastung erreicht werden. [3]

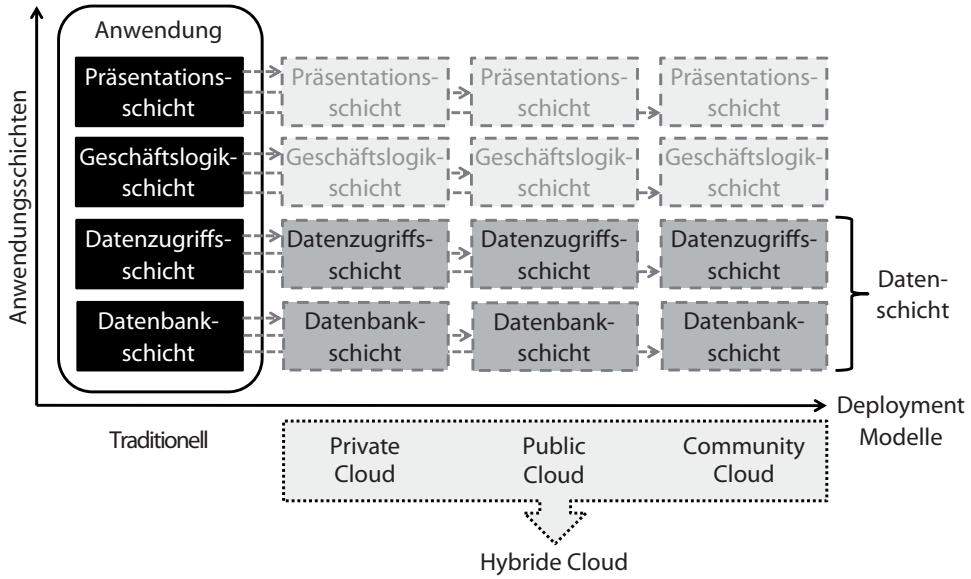

Abbildung 1 Hosting-Topologien für Cloud Anwendungen [4]

Um die Vorteile des Cloud Computing nutzen zu können, müssen bestehende lokale Anwendungen in die Cloud portiert oder neue Anwendungen speziell für die Cloud entwickelt werden. Bei der Betrachtung einer typischen Drei-Schichten-Architektur [5] von Anwendungen stellt sich die Frage, welchen Teil man in der Cloud laufen lassen möchte. Abbildung 1 zeigt eine Übersicht möglicher Verteilungen der Anwendungsschichten in die Cloud. Es besteht die Möglichkeit die gesamte Anwendung in die Cloud zu portieren oder aber einen Schnitt innerhalb einer Schicht oder zwischen zwei Schichten zu ziehen. Dieser Schnitt legt fest, welcher Teil der Anwendung in der Cloud läuft und welcher lokal bleibt. Auch wenn eine Anwendung komplett in die Cloud migriert werden soll, bietet es sich bei komplexen Anwendungen an, schrittweise vorzugehen.

Ein möglicher Schnitt ist die Trennung der Datenschicht in eine Datenzugriffsschicht (Data Access Layer, DAL) und eine Datenbankschicht (Database Layer, DBL). Damit könnte der DBL mit häufig sensiblen Daten lokal bleiben und die darüber liegenden Schichten inklusive DAL in die Cloud migriert werden, sofern sichergestellt ist, dass der sensible Teil der Daten nicht oder nur verschlüsselt (voll homomorphe Verschlüsselung [6]) bzw. anonymisiert oder pseudonymisiert [4] in die Public Cloud gelangt. Diese Trennung kann Kosten- und Elastizitätsvorteile bei geplanten und ungeplanten hohen

Lasten bringen, sofern die lokal verbliebene Datenbankschicht ebenfalls mit den Lastspitzen zuverlässig umgehen kann.

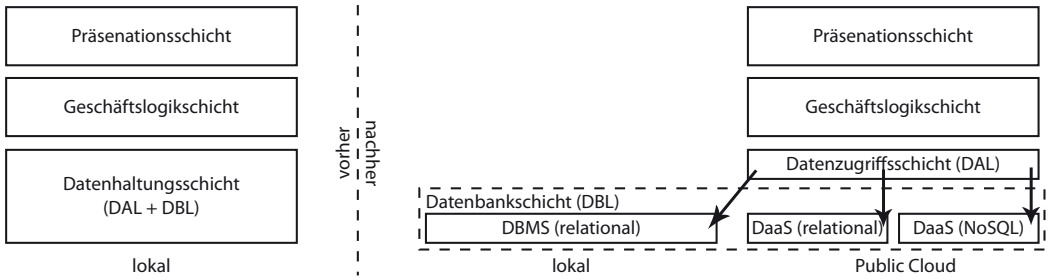

Abbildung 2 Szenario einer Cloud Datenbankmigration

Ein komplexeres Szenario zur Migration des lokalen DBL in die Cloud ist in Abbildung 2 dargestellt. Vor der Migration wurden die Daten in diesem Beispiel in einem lokalen relationalem Datenbank Managementsystem (RDBMS) gehalten. Während der Cloud Datenmigration werden die Daten aus dem lokalen RDBMS auf verschiedene Datenspeicher aufgeteilt. Sensible Daten wie Kreditkarteninformationen und persönliche Daten verbleiben im lokalen RDBMS. Die restlichen Daten werden verteilt in einen nichtrelationalen Cloud Datenbankdienst (NoSQL) und relationalen Cloud Datenbankdienst (Polyglot Persistence [7]). Diese Aufteilung ermöglicht eine bessere Skalierbarkeit, schnelleren Zugriff und eine höhere Verfügbarkeit durch die Aufweichung der Konsistenzanforderungen in NoSQL Datenspeichern (siehe CAP-Theorem [8]). Die Aufteilung der Daten wird dabei so vorgenommen, dass Tabellen mit wenigen Einträgen und seteneren Lese- und Schreibzugriffen in einem relationalen Cloud Datenbankdienst gehalten werden, was einen geringeren Migrationsaufwand bedeutet. Sehr große Tabellen mit sehr häufigen Lese- und Schreibzugriffen werden in einem NoSQL Cloud Datenbankdienst gehalten.

Auch umgekehrt ist es vorstellbar, dass der DBL in die Cloud ausgelagert wird und der restliche Teil der Anwendung weiter lokal betrieben wird. Dieser Ansatz liegt dieser Arbeit zu Grunde und ist zum Beispiel für Anwendungen relevant in denen eine höhere Latenz beim Zugriff auf Daten kein großes Problem darstellt. Dabei müssen die Daten nicht zwangsläufig in eine öffentliche Cloud ausgelagert werden, sondern können z.B. in eine private oder Community Cloud migriert werden, um den Sicherheits- und Datenschutzanforderungen in Bezug auf sensible Daten gerecht zu werden.

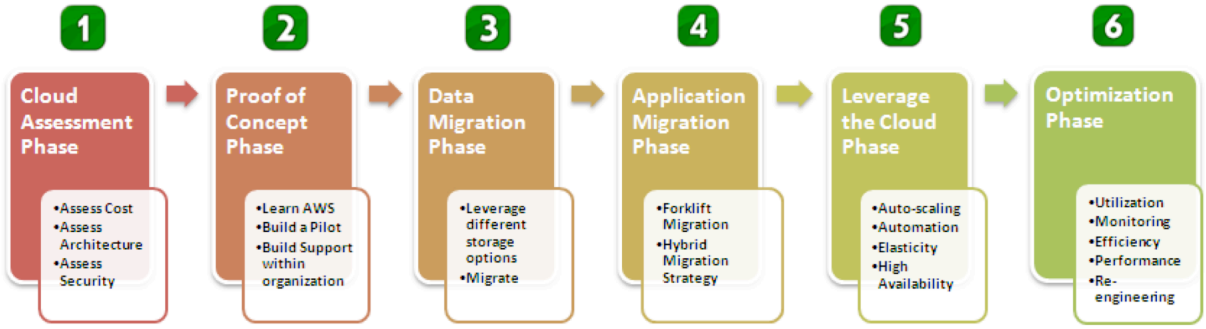

Abbildung 3 Phasengetriebene Herangehensweise der Cloud Migration von AWS [9]

Da in der Regel aber nicht nur der DBL einer Anwendung in die Cloud migriert wird [9], sondern häufig die gesamte Anwendung, kann die hier entwickelte Methodik ebenfalls als ein Schritt innerhalb der gesamten Cloud Anwendungsmigration, wie in Abbildung 3 gezeigt, gesehen werden. Jedoch ist der Fokus dieser Arbeit die alleinige Migration des

DBL in die Cloud. Dass dies selbst ohne Migration der gesamten Anwendung sinnvoll ist, zeigen die Szenarien in Abschnitt 4.1. Weiterhin geht es nicht nur um eine bloße Verschiebung von virtuellen Maschinen in eine IaaS Cloud. Mit neuen Cloud Diensten wie Database-as-a-Service (DaaS) ist es nun möglich einzelne Schichten einer Anwendung in die Cloud zu verschieben und somit auch allein den DBL zu migrieren. Diese Arbeit umfasst damit auch und vor allem die Migration von einem lokalen oder entfernten DBL in einen Cloud-spezifischen Datendienst.

Abhängig von z.B. der Vertraulichkeit der Daten muss zusätzlich eine passendes Cloud Deployment Model [10] gewählt werden, denn sehr sensible Daten dürfen zum Beispiel nicht ohne Weiteres in einer Public Cloud abgelegt werden.

Da es bisher noch wenig Unterstützung bei der Datenmigration in die Cloud gibt, führt diese Arbeit eine ganzheitliche Methodik ein um Daten in die Cloud zu migrieren. Dabei werden verschiedene existierende Migrationsmethoden betrachtet, Anwendungsszenarien für die Datenmigration identifiziert und in Zusammenhang mit Cloud Data Hosting Lösungen [11] und Cloud Data Patterns [12] gestellt. Zusätzlich wird herausgestellt, welche Auswirkungen die Migration des DBL auf die darüber liegenden Schichten hat.

Abschließend soll eine Anwendung erstellt werden, die bei der Datenmigration sowohl im Entscheidungsprozess, als auch beim eigentlichen Migrieren der Daten und den eventuell erforderlichen Anpassungen der Anwendungsarchitektur unterstützt.

Menzel et al. [13] referenziert das Thema dieser Arbeit: „Future work also includes extending the support to more application types and to multiple, interrelated components. Besides, support for a persistence layer and system trade-offs such as the CAP theorem or security vs. latency as well as a wider process loop that begins with an initial Cloud decision are currently studied."

1.2 Problemstellung und Zielsetzung

Bisher gibt es noch wenig Unterstützung für die Migration von Daten in die Cloud und zwischen Cloud Diensten. Es gibt zwar einzelne Anbieter wie Informatica Cloud [14] und Anwendungen wie Apex Data Loader [15] die Daten in bestimmte Cloud Dienste kopieren und mit lokalen und entfernten Datenquellen synchronisieren können, es fehlt aber an einer Methodik die hilft das eigene Problem zu klassifizieren und mögliche Migrationsstrategien vorzuschlagen. Außerdem fehlt eine Übersicht an Datenmigrationsszenarien mit Cloud Bezug sowie eine Taxonomie dazu.

Das Ziel dieser Arbeit ist eine ganzheitliche Methodik zur Migration der Datenschicht in die Cloud zu entwickeln. Dabei sollen aktuelle Datenmigrationstechniken, verschiedene Migrationsszenarien, Cloud Data Hosting Lösungen und Cloud Data Patterns in Betracht gezogen werden. Die Methodik muss außerdem erweiterbar sein für Anforderungen wie z.B. Compliance oder Datensicherheit und Hinweise auf die Anpassung von Anwendungsschichten oberhalb das DBL bieten.

Die entwickelte ganzheitliche Methodik unterstützt ebenfalls bei der Entscheidungsfindung eines geeigneten Migrationsszenarios sowie der Datenmigration für ausgewählte Migrationsszenarien. Dies wird in einer prototypischen Realisierung umgesetzt und anschließend evaluiert.

1.3 Abgrenzung

Die zu entwickelnde ganzheitliche Methodik dieser Arbeit beschränkt sich auf die Migration von Daten in die Cloud. Dabei soll sowohl die Taxonomie der Datenmigrationsszenarien, als auch die ganzheitliche Methodik, die Migration zwischen verschiedenen Cloud Deployment Models berücksichtigen. Die prototypische Implementierung beschränkt sich auf die Unterstützung ausgewählter Migrationsszenarien und deren Evaluierung an Beispielen. Die Migration der Präsentations- oder Anwendungsschicht sind nicht Bestandteil dieser Arbeit.

Außerdem wird die Anpassung des DAL und höheren Anwendungsschichten nur eine geringe Rolle spielen und sich auf die Beschreibung der im Rahmen dieser Arbeit gefundenen Herausforderungen und Auswirkungen auf andere Anwendungsschichten beschränken.

Diese Arbeit beschäftigt sich weiter nicht mit der Integration von Anwendungen und deren Daten, hier sei auf Linthicum [16] verwiesen.

Das zu entwickelnde Migrationstool soll so erweiterbar sein, dass in Zukunft möglichst viele Ziel-, viele Quelldatensysteme und Cloud Datenspeicher unterstützt werden können. Die prototypische Implementierung im Rahmen dieser Arbeit wird sich auf sechs verschiedene Cloud Datenspeicher (DB-Installation innerhalb Amazon EC2, Amazon RDS, Amazon SimpleDB, Google Cloud SQL, Google App Engine Datastore, SQL Azure) und zwei Quelldatensystems (MySQL, CSV Datei) beschränken, um zu zeigen, dass das Konzept prinzipiell funktioniert.

Es wird weiter davon ausgegangen, dass die Entscheidung, eine komplette Anwendung oder zumindest die Daten einer Anwendung in die Cloud zu migrieren, schon getroffen wurde. Damit wird in der ganzheitlichen Datenmigrationsmethode nicht auf die generellen Vor- und Nachteile oder Aspekte des Cloud Computings im Vergleich zu lokal betriebenen Anwendungen eingegangen. Es ist dem Autor bewusst, dass diese Einschränkung zum Teil zu groß ist, da die passende Cloud Data Hosting Lösung mit ihren verbundenen Kosten evtl. gegen eine Entscheidung der Migration einer Anwendung in die Cloud führen kann. Dennoch wird auf Grund des Fokus dieser Arbeit auf die Migration des DBL darauf verzichtet.

Die Migrationsmethodik beinhaltet keine Aspekte des Outsourcings der Migrationsaufgaben. Um eine Migration ganzheitlich abzubilden, müssen alle Schritte beachtet werden. Somit spielt z.B. die Identifikation von Dienstleistern, die bei der Migration helfen oder sie ganz übernehmen können, wie von Laszewski in [17] beschrieben, in dieser Methodik keine Rolle.

Auf regulatorische Richtlinien zur Migration von u.a. urheberrechtlich geschützten oder sensiblen Daten in die Cloud wird nicht eingegangen und auf Höllwarth [18] verwiesen.

Für Anforderungen an Datensicherheit, Compliance und Auditfähigkeit während der Datenmigration wird auf ein Whitepaper von Netspective Communications LLC [19] verwiesen und in dieser Arbeit nicht weiter eingegangen. Die entstehende Migrationsmethodik und das Datenmigrations-Tool sind aber dahingehend erweiterbar.

1.4 Gliederung

In diesem Abschnitt wird die Gliederung dieser Arbeit beschrieben, welche sich am wissenschaftlichen Vorgehen im Rahmen dieser Arbeit orientiert.

Nach der Motivation und Aufgabenstellung dieser Arbeit in den Abschnitten 1.1 und 1.2 und einer Abgrenzung zu verwandten Themen in Abschnitt 1.3 folgt die Darstellung des aktuellen Kenntnisstandes auf dem Themengebiet der Cloud Datenmigration in Kapitel 2. Hier werden die aktuellen Arbeiten beschrieben, gezeigt auf welche Informationen zurückgegriffen wird und der Beitrag dieser Arbeit zum Thema Cloud Datenmigration zum aktuellen Stand in Beziehung gesetzt und positioniert.

In Kapitel 3 wird auf grundsätzliche Themen eingegangen und grundlegende Begriffe definiert. Diese stellen die Basis für diese Arbeit dar und auf diese wird später zurückgegriffen.

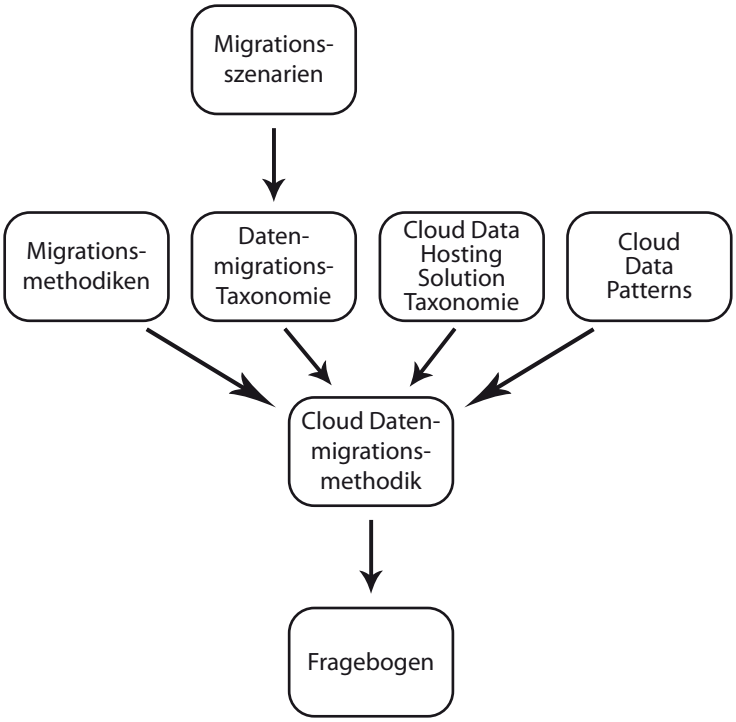

Abbildung 4 Theoretischer Teil der Arbeit

Der theoretische Teil der Arbeit wird in den Kapiteln 2 bis 4 behandelt. Wie aus Abbildung 4 zu entnehmen ist, werden in die zu entwickelnde Cloud Datenmigrationsmethodik (Abschnitt 4.7) verschiedene existierende Migrationsmethoden (Abschnitt 2.1 und 2.2), Migrationsszenarien (Abschnitt 4.1), Cloud Data Hosting Lösungen (Abschnitt 4.3) und Cloud Data Patterns (Abschnitt 4.4) einfließen. Aus den Migrationsszenarien wird zusätzlich noch eine Taxonomie (Abschnitt 4.2) hervorgehen, welche die verschiedenen Migrationsszenarien kategorisiert. Alle dazu benötigten grundlegenden Konzepte werden in Kapitel 3 beschrieben. Wie mit den verschiedenen Typen von Daten wie Anwendungs- oder Konfigurationsdaten während der Migration umgegangen wird, wird in Abschnitt 4.5 behandelt. Abschnitt 4.6 enthält zudem Auswirkungen der Migration der Datenbankschicht auf darüber liegende Schichten wie den DAL oder die Anwendungsschicht. Diese gesamten Vorarbeiten fließen dann in die ganzheitliche Cloud Datenmigrationsmethodik in Abschnitt 4.7 ein.

Zur Vorbereitung des Cloud Datenmigration-Tools, welches auch ein Entscheidungs-Unterstützungssystem beinhaltet, werden aus der entstandenen Methodik Fragebögen (Abschnitt 4.8) entwickelt. Diese Fragebögen haben zum Ziel, das Migrationsszenario zu identifizieren und Cloud Data Hosting Solutions einzuschränken.

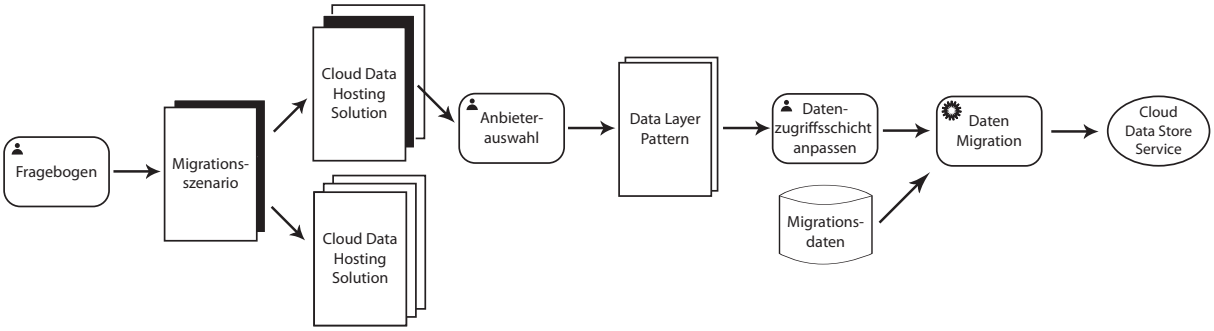

Abbildung 5 Praktischer Teil der Arbeit

Im praktischen Teil der Arbeit werden die entwickelten Fragebögen in ein Entscheidungs-Unterstützungssystem eingebaut und mit einer Liste von Anbietern und deren Lösungen kombiniert. Sofern eine Lösung eines Anbieters bestimmte Anforderungen nicht erfüllt, werden zusätzlich noch Data Layer Pattern vorgeschlagen um die Funktion nachzubilden. Zusätzlich wird eine Hilfestellung bei der Anpassung der Datenzugriffsschicht gegeben. Schließlich werden die Daten aus einem Quellsystem in einen Cloud Daten Dienst migriert.

Die Entwicklung des Cloud Datenmigrations-Tools wird in Kapitel 5 beschrieben und die Anwendung des Tools in konkreten Anwendungsfällen wird im darauf folgenden Kapitel 6 protokolliert und evaluiert.

Abschließend werden die Ergebnisse der Arbeit in Kapitel 7 zusammengefasst und ein Ausblick des Themenbereichs Cloud Datenmigration sowie Anknüpfungspunkte für weitere Arbeiten gegeben.

2 Kenntnisstand

Um zu zeigen worin der wissenschaftliche Beitrag dieser Arbeit besteht, wird zuerst beschrieben welche aktuellen Arbeiten auf dem Gebiet der Cloud Datenmigration bestehen und wie sie in dieser Arbeit genutzt werden. Dabei wird auch erläutert worin der eigene Beitrag dieser Arbeit besteht und wie sich dieser in Bezug auf bestehende Arbeiten positioniert.

2.1 Methoden von Cloud Anbietern zur Datenmigration

Exemplarisch für Cloud Anbieter werden im Folgenden die Datenmigrationsstrategien von Amazon Web Services, Microsoft Azure, Google App Engine und Salesforce näher betrachtet. Die dabei berücksichtigten Datenspeicherdienste und Migrationsanleitungen der einzelnen Anbieter werden in die Anbieterauswahl der Datenmigrationsmethodik, die Data Layer Patterns Auswahl und in das Cloud Datenmigrations-Tool einfließen.

2.1.1 Amazon Web Services

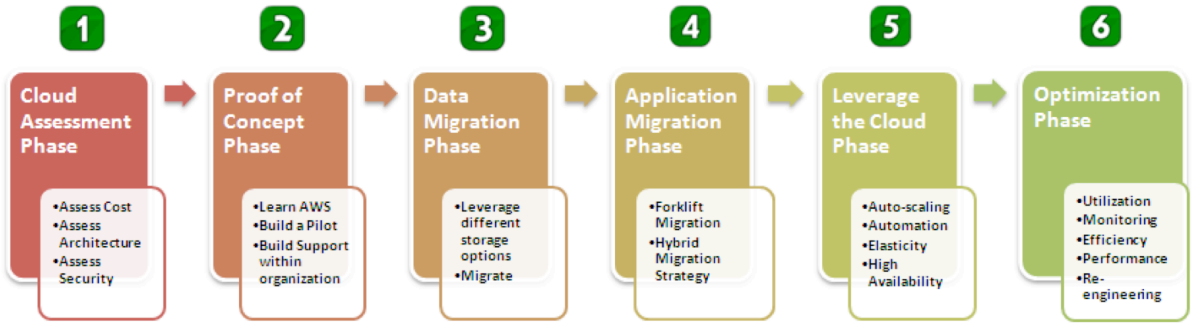

Abbildung 6 Amazons 6-Phasen Modell zur Cloud Migration [9]

Amazon Web Services (AWS) hat eine phasen-getriebene Herangehensweise für die Cloud Migration entwickelt [9], die in Abbildung 6 dargestellt ist. Die Datenmigration wird hier als einzelne Phase aufgeführt, in der zum einen ein Cloud Datenspeicher ausgewählt wird und zum anderen die Daten migriert werden. Amazon bietet dafür eine Übersicht von AWS Datendiensten sowie eine Beschreibung für welche Klassen von Daten sich die einzelnen Dienste eignen. Der Ablauf der Datenmigration wird sehr kurz beschrieben, ohne technische Hilfestellungen. Für einzelne AWS Dienste wie RDS gibt es ausführliche technische Migrationsleitfäden um Daten einer MySQL 5.1 Datenbank nach Amazon RDS mit MySQL Engine zu migrieren [20].

	Amazon S3 + CloudFront	Amazon EC2 Ephemeral Store	Amazon EBS	Amazon SimpleDB	Amazon RDS
Ideal for	Storing large write-once, read-many types of objects, Static Content Distribution	Storing non-persistent transient updates	Off-instance persistent storage for any kind of data,	Query-able light-weight attribute data	Storing and querying structured relational and referential data
Ideal examples	Media files, audio, video, images, Backups, archives, versioning	Config data, scratch files, TempDB	Clusters, boot data, Log or data of commercial RDBMS like Oracle, DB2	Querying, Indexing Mapping, tagging, click-stream logs, metadata, Configuration, catalogs	Web apps, Complex transactional systems, inventory management and order fulfillment systems
Not recommended for	Querying, Searching	Storing database logs or backups, customer data	Static data, Web-facing content, key-value data	Complex joins or transactions, BLOBs Relational, Typed data	Clusters
Not recommended examples	Database, File Systems	Shared drives, Sensitive data	Content Distribution	OLTP, DW cube rollups	Clustered DB, Simple lookups

Abbildung 7 Speicherdienste in der AWS Cloud [9]

Eine Beschreibung von Datenmigrationsszenarien erfolgt nicht über die Tabelle aus Abbildung 7 hinaus. Im Januar 2012 hat Amazon zusätzlich DynamoDB [21] veröffentlicht. Dieser NoSQL Dienst ist auf konsistente und schnelle Performance sowie unendliche Skalierbarkeit ausgelegt und gleicht ansonsten dem SimpleDB Dienst. Im August 2012 wurde ein weiterer Datendienst spezielle für Backups und Archivierung namens Amazon Glacier [22] veröffentlicht, welcher Daten sehr günstig archiviert und um mehrere Stunden zeitversetzt auf Anfrage wieder verfügbar macht.

AWS beschreibt zusätzlich die in Abbildung 8 dargestellten Cloud Anwendungs-Migrationsszenarien, welche für eine Migration in die Cloud in Frage kommen:

- Web Application
- Batch Processing Pipeline
- Backend Processing Workflow

Diese enthalten auch kurze Informationen zum prinzipiellen Vorgehen bei der Migration. [23–25]

Scenario Name	Solution	Use case	Motivation For migration	Additional Benefits	Services Used
Company A	Web Application	Marketing and collaboration Web site	Scalability + Elasticity	Auto Scaling, pro-active event based scaling	EC2, S3, EBS, SimpleDB, AS, ELB, CW, RDS
Company B	Batch processing pipeline	Digital Asset Management Solution	Faster time to market	Automation and improved development productivity	EC2, EBS, S3, SQS
Company C	Backend processing workflow	Claims Processing System	Lower TCO, Redundancy	Business continuity and Overflow-protection	EC2, S3, EBS, AS, SQS, IE

Abbildung 8 Cloud Migrationsszenarien [9]

Im letzten Schritt des 6-Phasen Modells von Amazon werden unter dem Punkt *Decompose your Relational database* folgende weitere Optimierungen vorgeschlagen um den DBL besser skalierbar zu machen [9]:

- *Move large blob object and media files to Amazon S3 and store a pointer (S3 key) in your existing database*
- *Move associated meta-data or catalogs to Amazon SimpleDB*
- *Keep only the data that is absolutely needed (joins) in the relational database*
- *Move all relational data into Amazon RDS so you have the flexibility of being able to scale your database compute and storage resources with an API call only when you need it*
- *Offload all the read load to multiple Read Replicas (Slaves)*
- *Shard (or partition) the data based on item IDs or names*

Die hier vorgestellten AWS Dienste werden in die Anbieterauswahl der ganzheitlichen Methodik einfließen. Aus den Cloud Migrationsszenarien von AWS werden passende Datenmigrationsszenarien abgeleitet sowie andere gefundene Datenmigrationsszenarien zugeordnet. Die Optimierungen bezüglich technischer Restriktionen und einer besseren Skalierbarkeit des DBL werden ebenfalls in die ganzheitliche Methodik eingehen und erweitert. Im Gegensatz zu dem in dieser Arbeit verfolgten Ansatz von Laszewski [17] berücksichtigt die von Amazon vorgeschlagene Migrationsmethodik keine explizite Trennung der Test und Deployment Phase und beinhaltet auch keine Post-Production Support Phase.

2.1.2 Microsoft Azure

Microsoft stellt ähnlich wie AWS generische Szenarien für mögliche Anwendungen in der Cloud auf. Darunter befinden sich die folgenden acht Anwendungstypen, welche für den Betrieb in der Cloud in Frage kommen [26–33]:

- SaaS Anwendungen
- Hochskalierbare Webseiten
- Enterprise Anwendungen
- Business Intelligence (BI) und Datawarehouse Anwendungen
- Soziale oder kundenorientierte Anwendungen
- Soziale (online) Spiele
- Mobile Apps
- High Performance Computing (HPC) oder parallele Anwendungen

Bei der Anpassung der Datenzugriffsschicht schlägt Microsoft vor, bei bestehenden Anwendungen die auf Microsoft SQL Server beruhen, lediglich die Datenbank-Verbindungskennung an den SQL Database Service anzupassen und zusätzlich bisher lokal gespeicherte Dateien im Windows Azure Blob Storage zu speichern. [34]

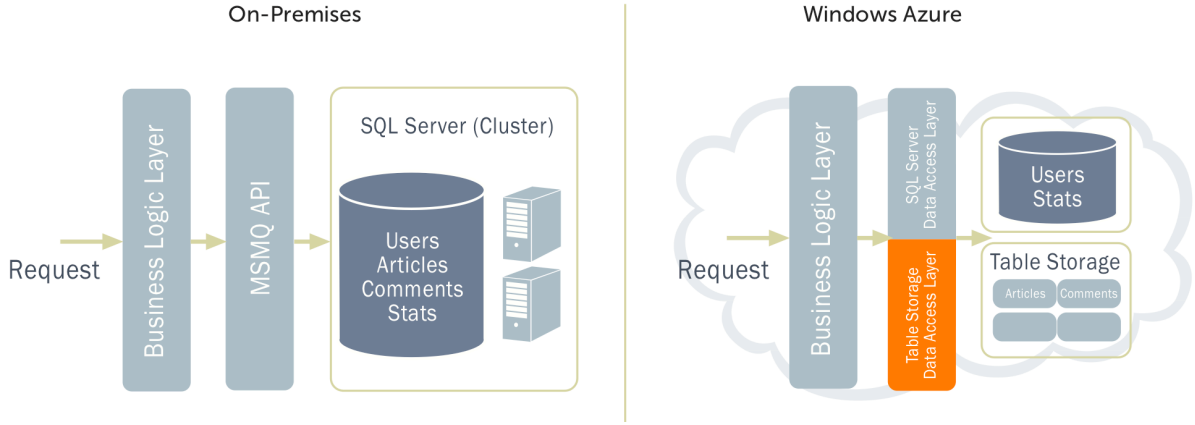

Abbildung 9 Microsoft SQL Server Migration [35]

In einem weiteren Migrationsleitfaden wird empfohlen große Tabellen mit mehreren Millionen Einträgen und häufigen Lese- und Schreibanfragen entweder auf mehrere SQL Database Instanzen zu verteilen oder sie in den Microsoft Azure Tables Dienst zu verschieben. Bevorzugt wird das Verschieben von großen Datenmengen, wie in Abbildung 9 gezeigt, in den Microsoft Azure Tables Dienst. Dieser ist für Datenmengen bis zu 100 TB ausgelegt. Die Verteilung von Datensätzen auf mehrere Datenbankinstanzen hingegen ist komplex und zeitaufwendig. Andere Tabellen mit wenigen Einträgen und seteneren Zugriffen können in einer SQL Database Instanz verbleiben. [35]

Microsoft gibt zusätzlich Empfehlungen zur Segmentierung von Daten, allerdings sind diese nicht Cloud-spezifisch. Weiterhin wird empfohlen Kreditkartendaten nicht in der Windows Azure Cloud zu speichern, da dies mit dem Payment Card Industry Data Security Standard (PCI DSS) nicht vereinbar ist. [36]

Ähnlich wie AWS gibt auch Microsoft eine ausführliche technische Schritt-für-Schritt Anleitung um lokale Microsoft SQL Server Datenbanken in den SQL Database Service zu migrieren. Die Migration erfolgt in zwei Schritten, zuerst wird eine Wizard-gestützte Schema-Migration initiiert und danach eine Kommandozeilen-getriebene Datenmigration. Es ist nicht möglich ein lokales Backup in den SQL Database Service direkt einzuspielen. SQL Database ist außerdem nicht voll kompatibel zu Microsoft SQL Server, da nicht alle TSQL Erweiterungen unterstützt werden. Weiterhin gibt es Größenbeschränkungen, so darf es z.B. nur 150 Datenbanken pro SQL Database Server geben und jede Datenbank darf maximal 150 GB fassen [37]. Microsoft (Nethi et al.) empfiehlt die Datenmigration

in mehrere Aufgabenpakete zu unterteilen und diese parallel auszuführen. Dabei müssen die Fremdschlüsselbeziehungen eingehalten oder die Constraint-Überprüfung muss temporär abgeschaltet werden. [38]

Neben dem Migrationsleitfaden gibt es einen SQL Azure Migration Wizard der bei der Migration hilft und Kompatibilitätsprobleme identifiziert [39], eine Schritt-für-Schritt Anleitung inkl. Migrationsassistent für die Migration von Microsoft Access nach SQL Database [40] und den Synchronisationsdienst SQL Data Sync [41].

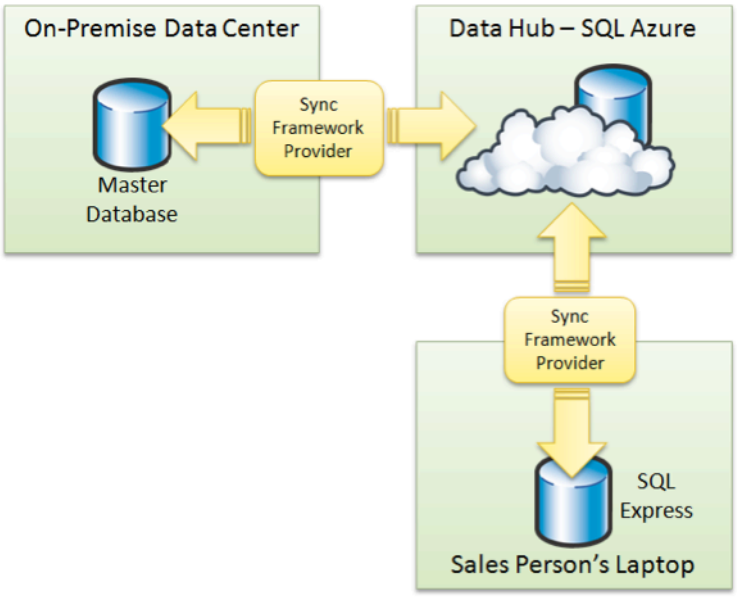

Abbildung 10 Konzeptuelle Ansicht des Daten Hub Szenarios [42]

Als spezielle Szenarien für die alleinige Migration der Daten nach SQL Database unter Verbleib der Anwendung im lokalen Rechenzentrum, sieht Microsoft Webanwendungen, Abteilungs- oder Arbeitsgruppen-Anwendungen und Daten Hubs. Unter Webanwendungen kann man z.B. Webshops oder Dokumentveraltungssysteme fassen. Abteilungs- oder Arbeitsgruppen-Anwendungen sind kleinere Anwendungen die nur von einer Abteilung für einen speziellen Zweck verwendet werden. Sie laufen zum Teil auf lokalen Rechnern von Mitarbeitern und bauen auf Microsoft Access, Microsoft SQL Server Express oder anderen SQL Servern auf. Bei diesen ersten beiden Szenarien werden allerdings die Performanceprobleme und Anwendungskomplexität durch die entstehende Latenz nach der Migration aufgezeigt. Im Daten Hub Beispiel, welches in Abbildung 10 gezeigt wird, werden z.B. Kundendaten und Preislisten auf den Laptops von Vertriebsmitarbeitern gespiegelt. Hier spielt die erhöhte Latenz eine untergeordnete Rolle, da die lokalen Datenbanken lediglich regelmäßig im Hintergrund mit einer SQL Database Datenbank synchronisiert werden und die Zugriffe der Anwendungen selbst auf die Kunden- und Preislistendaten stets lokal erfolgen. [42]

Weitere vier allgemeine Szenarien für die Cloud-Migration werden von Cunningham in [36] beschrieben und finden auch in dieser Arbeit Verwendung:

1. An/Aus: Die Anwendung wird nur zu bestimmten vordefinierten Zeiten benötigt.
2. Schnelles Wachstum: Der schnelle Anstieg an Ressourcen kann nicht durch Hardwarekäufe kompensiert werden.
3. Vorhersehbare Variabilität: Zyklische, wie Saisonale Schwankungen erlauben eine optimale Auslastung der Rechen- und Speicherressourcen.

4. Unvorhersehbare Variabilität: Ohne rechtzeitige Ankündigung steigt plötzlich die Nutzung einer Anwendung.

Redkar et al. [43] greifen die Ansätze von Nethi et al. [38] auf und erweitern sie um Hinweise zur Übertragung großer Datenmengen in die Cloud. Sie schlagen die Ausnutzung von Parallelität oder das Aufsuchen von Technologiepartnern vor, die eine bessere Netzwerkverbindung in das Rechenzentrum des jeweiligen Cloud Anbieters haben. Als Anwendungsszenarien für den SQL Database Service wird u.a. auch Datenbankkonsolidierung (Schatten-IT), die Speicherlösung für Windows Azure Compute und geographische Replikation von Daten genannt. Als Datenquellen für die Migration nach SQL Database mittels SQL Server Migration Assistant (SSMA) oder SQL Azure Migration Wizard von Codeplex wird Microsoft Access, MySQL, Microsoft SQL Server, IBM DB2, Oracle und Sybase genannt. Neben diesen Datenbanken können auch fast alle anderen relationalen Datenbanken nach SQL Database migriert werden, indem sie zuerst nach Microsoft SQL Server und von dort nach SQL Database migriert werden.

Die vorgestellten allgemeinen Cloud Anwendungs-Migrationsszenarien und Cloud Daten-Migrationsszenarien werden in verallgemeinerter Form in die Sammlung von Migrationsszenarien eingehen. Die technischen Hinweise zur Datenmigration werden sowohl in verallgemeinerter Form als auch in spezieller Form für die spezifische Migration von einem unterstützten Quellsystem in ein unterstütztes Zielsystem in die ganzheitliche Methodik eingehen.

2.1.3 Google App Engine

Google App Engine bietet ein Tool namens Bulk Loader welches sowohl den Import von CSV- und XML-Dateien in den App Engine Data Store, als auch den Export von Daten aus dem App Engine Data Store in CSV-, XML- oder Textdateien ermöglicht. Transformationen der Daten währen des Imports können in Konfigurationsdateien deklariert oder programmatisch festgelegt werden. [44], [45]

Um Daten innerhalb eines App Engine Master/Slave Data Stores zu einem App Engine High Replication Datastore zu migrieren, bietet Google die Möglichkeit eine bestehende Anwendung zu duplizieren und über einen online Migrations-Assistenten die Daten mit fast uneingeschränkter Anwendungsverfügbarkeit zu migrieren [46].

Google unterstützt App Engine Nutzer auch bei der Auswahl des passenden Datenspeichers, beziehungsweise dessen Konfiguration [47] oder beim Verschieben von Daten von einer Anwendung zur nächsten [48].

Die von Google vorgestellten Möglichkeiten zur Transformation von Daten während des Imports werden in verallgemeinerter Form in die ganzheitliche Methodik einfließen.

2.1.4 Salesforce

Salesforce [49] bietet verschiedene Wege Daten sowohl aus CRM spezifischen Quellen wie Kontakte aus ACT! oder Microsoft Outlook, aus Eigenentwicklungen, als auch aus CSV Dateien zu importieren. Der Import kann über die Weboberfläche von Salesforce, mehrere APIs, einen Excel Connector oder eine Windows Desktop Anwendung namens Apex Data Loader geschehen. Zusätzlich gibt es Partner wie Informatica Cloud [14], die bei Datenmigrationen und Synchronisationen unterstützen können. [50] [51]

Eine Fünf-Schritt-Anleitung hilft beim Importieren von Daten aus externen Systemen. Die Schritte beinhalten der Reihe nach Auswahl der zu importierenden Daten, die Auswahl des Migrationswerkzeugs, Datenbereinigung, Importvorbereitung und Test, sowie Import und Validierung des Datenimports. [50]

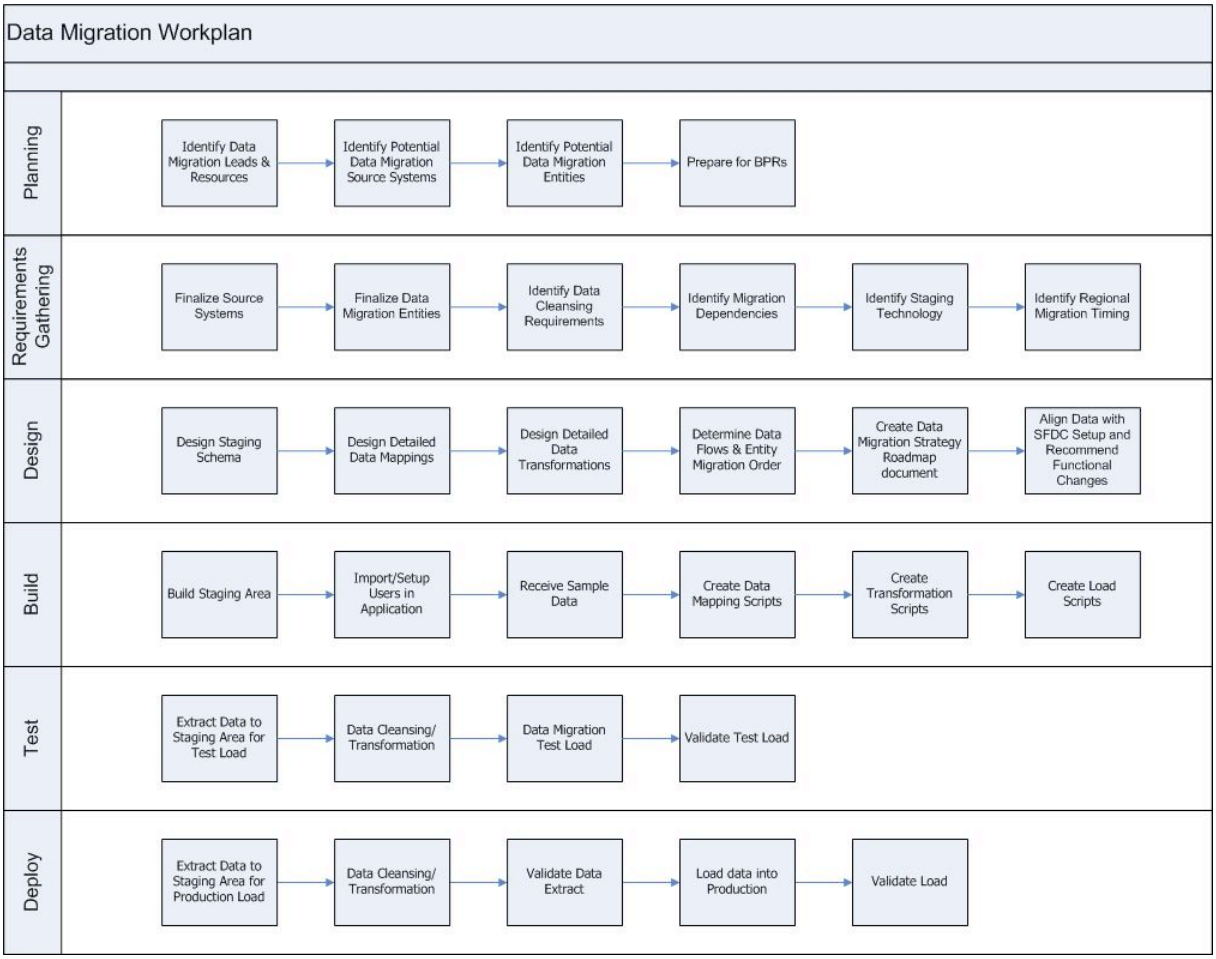

Abbildung 11 Salesforce Datenmigrationsplan [52]

Eine detailliertere beispielhafte Salesforce Methodik zur Datenmigration ist in Abbildung 11 zu sehen und in [53] dokumentiert.

Salesforces Fünf-Schritt-Anleitung ist zum Großteil anbieterunabhängig und ähnelt in vielen Schritten der umfassenderen Methodik von Laszewski [17]. Salesforces Methodik wird, um Salesforce spezifische Werkzeuge bereinigt, in die ganzheitliche Methodik einfließen. Die Migration von CRM Daten nach Salesforce wird allerdings nicht Bestandteil des Migrations-Tools werden, da dem Autor keine repräsentativen (Test)Daten aus bestehenden on-premise CRM Systemen zur Verfügung stehen.

2.2 Empfehlungen Anbieter-unabhängiger Quellen zur Datenmigration

Neben den Migrationsleitfäden der Cloud-Anbieter existieren unabhängige Quellen, die Empfehlungen für die Migration von Daten in die Cloud bieten. Deren Inhalte werden in diesem Abschnitt zusammengefasst. Dabei werden sowohl spezielle Cloud Datenmigrationsmethoden einbezogen, als auch allgemeinere Cloud-Migrationsmethoden, allgemeine Daten-Migrationsmethoden und Software-Migrationsmethoden.

2.2.1 Migration im Allgemeinen

Der Migrationsleitfaden des CIO der Bundesregierung [54] definiert Migration wie folgt:

> *Eine Migration ist eine wesentliche Veränderung der vorhandenen Systemlandschaft oder eines beträchtlichen Teils derselben. Sie kann sich sowohl auf Hardware als auch auf Software beziehen.*

Dabei werden drei Arten von Migrationen unterschieden: Aktualisierung, fortführende Migration und ablösende Migration. Die Aktualisierung (engl. Update) bezeichnet einen einfachen Wechsel zur nächsten Version, der mit geringem Aufwand möglich ist. Die fortführende Migration (engl. Upgrade) hingegen beinhaltet grundlegende Änderungen am Produkt und hat somit Auswirkungen auf das Umfeld. Es kann auch notwendig sein Migrationswerkzeuge zu verwenden um z.B. Datenbestände zu transformieren. Die ablösende Migration beinhaltet einen Produktwechsel oder das Überspringen einer Produktgeneration und geht mit erheblichem Aufwand einher, da keine geeigneten Migrationswerkzeuge oder Dokumente existieren. Zusätzlich zu diesen drei Arten kann eine Migration auf zwei Wegen geschehen, zum einen durch die Stichtagsumstellung, zum anderen durch eine schrittweise Migration.

Neben den drei vorgestellten Migrationsarten spielt in dieser Arbeit noch die Migration ohne Versionssprung eine Rolle. Hier wird z.B. der Datenbestand eines lokal laufenden Datenbank Managementsystems (DBMS) auf ein DBMS mit gleicher Version, welches aber in der Cloud läuft, migriert. Auch die zwei verschiedenen Migrationswege werden in die allgemeine Cloud Datenmigrationsmethodik einfließen.

Zum technischen Vorgehen einer Migration sagt der Migrationsleitfaden nichts aus, ebenso wenig zur Datenmigration. Das Thema Cloud Migration wird im Kapitel „Zukunftsthemen der IT“ kurz beschrieben und deutet auf eine zukünftige „hybride Bundes-Cloud“ für Bundesbehörden hin.

2.2.2 Daten Migration im Allgemeinen

Reddy definiert Datenmigration wie folgt [55]:

> *It is the process of transferring data between storage types, formats, or computer systems.*

Zusätzlich ist die Datenmigration im Allgemeinen automatisiert und wird benötigt wenn ein Computersystem ausgetauscht oder aktualisiert wird oder wenn mehrere Computersysteme verschmelzen. Im Cloud Computing kommt noch die Migration von lokalen Daten zu in der Cloud gehosteten Daten hinzu. Beim Migrieren werden allgemein die Daten aus dem alten System dem neuen System zugeordnet, wobei ein Entwurf für die Extraktion, Transformation und Import (Extract-Transform-Load, ETL) erstellt wird. Nach der Migration müssen die Daten im neuen System verifiziert werden, um sicherzustellen, dass sie korrekt übersetzt wurden, vollständig sind und den Prozessen im neuen System genügen. Zusätzlich kann eine automatisierte oder manuelle Datenbereinigung während der Migration stattfinden. Reddy fasst in [55] zusammen, dass die Datenmigration aus den Phasen Design, Extraktion, Reinigung, Import und Verifikation besteht. Diese Phasen werden bei der in dieser Arbeit beschrieben Migrationsmethode mit beachtet, insbesondere der Schritt der Datenbereinigung wird wiederverwendet, da er nicht explizit in der umfassenderen Methodik von Laszewski [17] vorkommt.

Weiter kategorisiert Reddy Datenmigration in Speichermigration, Datenbankmigration, Anwendungsmigration, Geschäftsprozessmigration und digitale Datenerhaltung. Bei der

Speichermigration wird das physikalische Speichermedium austauscht um von Medien mit höherer Speicherdichte Gebrauch zu machen, ohne die Daten an sich zu verändern. Die Datenbankmigration beinhaltet das Kopieren von Daten aus einem DBMS in ein DBMS eines anderen Datenbankherstellers oder in ein DBMS mit einer neueren Version. Der klassische ETL-Prozess findet bei der Anwendungsmigration statt, wenn zum Beispiel ein Wechsel des Customer Relationship Management (CRM) oder Enterprise Resource Planning (ERP) Anbieters bevorsteht. Bei der Geschäftsprozessmigration kann es durch Anpassung oder Optimierung von Geschäftsprozessen notwendig sein, Daten von einem System in ein anderes zu migrieren damit der neue Geschäftsprozess auf diese Daten zugreifen kann. Die digitale Datenerhaltung kopiert physikalisch die Daten von einem Medium auf ein anderes und ist ähnlich der Speichermigration. Sie nutzt aber nicht Speichermedien mit höherer Speicherdichte, sondern gleichartige Speichermedien, da die Lesbarkeit der Daten auf dem alten Speichermedium gefährdet ist. Die Methodik dieser Arbeit fokussiert sich auf die Speichermigration im weiteren Sinne, sowie die Datenbankmigration.

Allgemein charakterisiert Reddy die Datenmigration durch vier Phasen, welche auch in die Migrationsmethode dieser Arbeit eingehen werden:

1. Analyse und Definition der Quelldatenstruktur
2. Analyse und Definition der Zieldatenstruktur
3. Zuordnung der Quell- auf die Zieldatenstruktur mit optionaler Datenbereinigung
4. Definition des automatisierten oder manuellen Migrationsprozesses

Morris hat in [56] vier goldene Regeln der Datenmigration aufgestellt, welche auch in die zu entwickelnde, gesamtheitliche Cloud Datenmigrationsmethode einfließen:

1. Datenmigration ist ein Business und kein technisches Problem.
2. Das Business weiß es am besten.
3. Keine Organisation braucht, will oder wird für perfekte Datenqualität zahlen.
4. Wenn man es nicht zählen kann, zählt es nicht.

Die Essenz dieser goldenen Regeln ist, dass das IT Personal oft nicht über die Bedeutung der zu migrierenden Daten Bescheid weiß und viel unnötigen Aufwand betreibt um die Daten möglichst perfekt zu migrieren.

Die Objektdatenbank Versant Object Database (VOD) bietet einen Mechanismus zur transparenten Migration von Objekten innerhalb eines Clusters auf verschiedene Nodes. Der physikalische Speicherort ist für den Client dabei transparent, er kennt nur eine eindeutige ID und adressiert die Objekte anhand dieser. Da Objektdatenbanken in dieser Arbeit eine untergeordnete Rolle spielen, wird dieser Ansatz nicht weiter verfolgt. Er ist auch nicht leicht anwendbar auf andere DBMS wie RDBMS. [57]

2.2.3 Cloud Migration im Allgemeinen

Tran et al. hat in [58] das Function Point Verfahren zur IT Projektkostenschätzung an Cloud Migrationsprojekte angepasst (Cloud Migration Point, CMP). Dazu wird u.a. auch eine Klassifikation der in die Cloud zu migrierenden Anwendung vorgenommen, welche in dieser Arbeit verwendet wird. Nach Tran et al. [58] gibt es eine vollständige und teilweise Migration in der Komponenten migriert und damit verändert, entfernt oder hinzugefügt werden können.

Mohan beschreibt in [59], dass eine Cloud Migration in fünf Ebenen geschehen kann: *„application, code, design, architecture, and usage“*. Neben diese fünf Ebenen läuft laut Mohan die Migration einer Enterprise-Anwendung wie folgt ab: $P \rightarrow P'_C + P'_I \rightarrow P'_{OFC} + P'_I$.

Dabei ist P die zu migrierende Anwendung, diese wird im ersten Schritt zweigeteilt, in einen Teil der in der Cloud läuft (P'_C) und einen Teil der lokal bleibt (P'_l). Wenn diese Phase abgeschlossen ist und der erste Teil der Anwendung in der Cloud läuft, wird dieser weiter für die Cloud optimiert (P'_{OFC}). Bei der kompletten Migration einer Anwendung ist P'_l leer, ansonsten handelt es sich um eine hybride Cloud Anwendung.

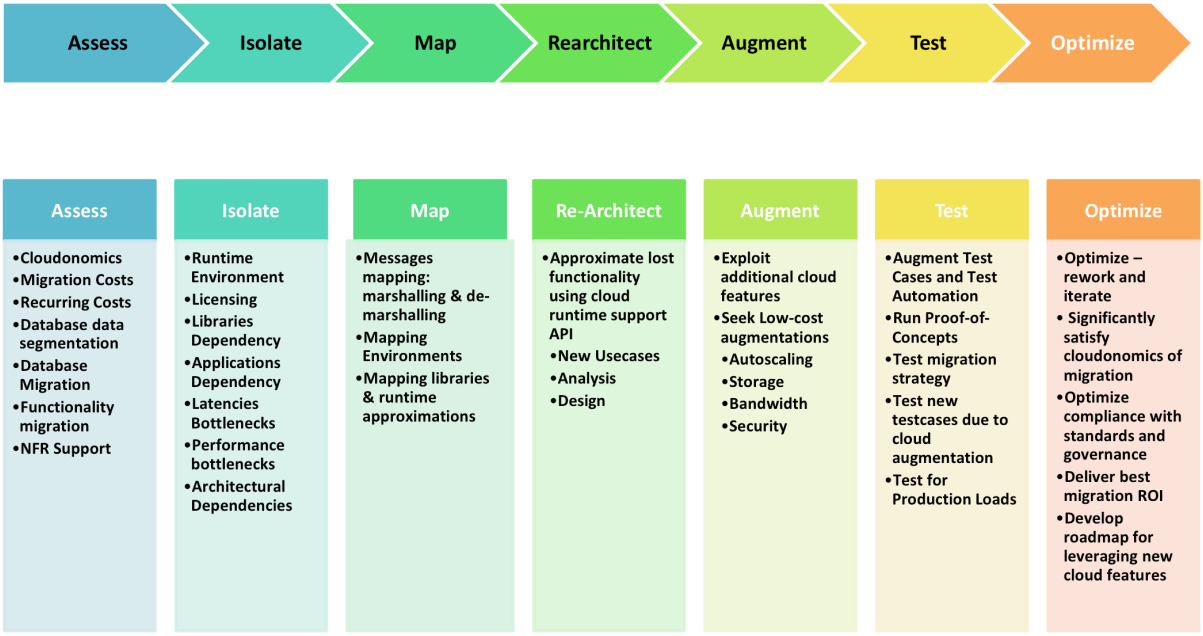

Abbildung 12 7-Schritt Migrations-Modell von Infosys Research [60]

Weiter hat Mohan ein generisches Sieben-Schritt-Modell, wie in Abbildung 12 dargestellt, für die Migration von Anwendungen in die Cloud entwickelt. Hier spielt bei der Datenmigration vor allem die Segmentierung der Daten eine Rolle, weiter wird auf die Datenmigration nicht eingegangen. In der ersten Phase wird analysiert ob eine Migration ökonomisch sinnvoll und technisch möglich ist, danach wird die zu migrierende Anwendung von systematischen und umgebenen Abhängigkeiten isoliert. Dabei kann festgestellt werden, wie komplex die Migration an sich wird. Im nächsten Schritt wird festgelegt welche Komponenten in die Cloud migriert werden und welche lokal bestehen bleiben. Damit steht auch fest welche architektonischen Änderungen mit der Migration einhergehen und welche Komponenten wie angepasst werden müssen. Im fünften Schritt werden die Architekturänderungen umgesetzt, wobei Funktionalitäten verloren gehen können. Der darauf folgende Schritt optimiert die Anwendung für die Cloud. Die letzten beiden Schritte umfassen dann den Test der Migration und darauf aufbauende weitere Optimierungen.

Mohan übt Kritik an Amazons Sechs-Phasen-Modell, da dies nur für die Migration der Anwendung in eine IaaS Umgebung mit minimaler Anpassung der zu migrierenden Anwendung ausgelegt ist. Allerdings spielt die Datenmigration in Mohans Modell eine geringe Rolle, sie legt mehr Wert auf architektonische Aspekte und Abhängigkeiten zu umliegenden System, welche auch in Form von Auswirkungen auf höhere Schickten in die Methodik dieser Arbeit einfließen werden.

Ähnlich zu Mohan [59] wird von Hajjat et al. [61] ein Modell zur Migration von Enterprise Anwendungen in eine hybride Cloud vorgestellt. Enterprise Anwendung bestehen oft aus mehreren Komponenten und der hybride Ansatz ermöglicht hier eine teilweise Migration in die Cloud auf Komponentenebene. Das Modell hilft bei der Auswahl von zu

migrierenden Komponenten im Enterprise Umfeld und unterstützt bei der Migration von Zugriffsmechanismen. Zuerst wird die Enterprise Anwendungsarchitektur mittels eines Graphen und einschränkenden Richtlinien beschrieben. Dazu werden alle Anwendungen und deren genutzten Komponenten sowie die Anzahl und Größe der Transaktionen zwischen den Komponenten und Anzahl an Servern, auf denen eine Komponente läuft ermittelt. Mit diesen Informationen kann dann entschieden werden, welche Auswirkungen in Form von Latenz und Traffic-Kosten die Migration einer Komponente auf die anderen Komponenten hat. In [61] wird explizit darauf hingewiesen, dass in dem vorgestellten Modell Datenbanken entweder ganz oder gar nicht migriert werden und nur auf einem Server laufen. Diese Einschränkung löst diese Arbeit und erweitert somit das in [61] vorgestellte Modell.

Laszewski et al. [17] haben in Kapitel Eins und Acht ihres Buches *Migration to the Cloud* fünf Szenarien zur Migration von Altsystemen (Legacy) in die Cloud herausgestellt: Neuschreiben bzw. Umstrukturieren, Plattformwechsel, Tool-gestützte automatisierte Konvertierung, Emulation durch Service-Oriented Architecture (SOA), Web Services oder Screen Scraper und Datenmodernisierung. Für diese Arbeit ist der Bereich Datenmodernisierung relevant. Hierunter verstehen Laszewski et al. die Migration des DBL in einen Cloud Datenbankservice um zusätzliche Funktionen wie Business Intelligence, Data Warehouse und Reporting zu ermöglichen. Laszewski et al. benennen dabei Vorteile wie schnelle und einfache Migration im Vergleich zur Migration des gesamten Altsystems und sehen Datenmodernisierung auch als ersten Schritt bei der Migration von kompletten Altsystem, da die Datenmodernisierung u.a. eine schnelle Informationsintegration ermöglicht. Als Nachteil wird die Abhängigkeit zu laufenden Altsystemen benannt, welche die zu migrierende Datenbank benötigen und somit einen Portierungsaufwand verursachen.

Armbrust et al. hat in [62] die folgenden Anwendungstypen als Katalysator für Cloud Computing herausgestellt:

- *Mobile interactive applications*
- *Parallel batch processing*
- *The rise of analytics, als Spezialfall von Stapelverarbeitung für Business Analytics*
- *Extension of compute-intensive desktop applications*

Für jeden dieser und weiterer Anwendungstypen (siehe Abschnitt 4.1.1) muss im konkreten Fall geprüft werden, ob eine der Elastizitäts-Charakteristiken von Cunningham [36] oder andere Gründe, wie z.B. geringere Kosten beim Betrieb oder höhere Verfügbarkeit, zutreffen. Dies ist notwendig um zu entscheiden ob eine Migration einer Anwendung in die Cloud überhaupt sinnvoll ist. In dieser Arbeit wird allerdings, wie in Abschnitt 1.3 beschrieben, davon ausgegangen, dass die Entscheidung zumindest den DBL einer Anwendung in die Cloud zu migrieren, schon getroffen wurde. Ob eine Cloud Datenmigration sinnvoll ist, muss anhand von Kriterien wie Sicherheit, Compliance, Wirtschaftlichkeit, Geschäftsnotwendigkeit und Kompatibilität zur bestehenden IT-Architektur entschieden werden. Weitere Hilfen dazu geben neben Armbrust et al. [62] und Cunningham [35], [36], u.a. auch Mohan [59].

2.2.4 Datenmigration in die Cloud

Tran et al. haben in [58] zwei Aufgabentypen bei der Datenmigration identifiziert, auf die in dieser Arbeit zurückgegriffen wird:

- *Query modification task: when a database changes in database type (e.g., MySQL to MSSQL), or database version, or from relational to NoSQL database, queries must be modified accordingly.* [58]
- *Data population task: Data in each table must be packaged and loaded into the new database.* [58]

Zusätzlich wird die Migration unterteilt in die Kategorien *„same type of relational database, same type of relational database but different versions, different types of relational databases, or relational to NoSQL database“* [58], welche in dieser Arbeit in der Taxonomie der Migrationsszenarien weiter verwendet werden.

Abbildung 13 Typischer Lebenszyklus eines Migrationsprojekts [17]

Ähnlich zu Tran et al. [58] haben Laszewski et al. [17] in Kapitel Zwei spezielle Aufgaben für die Datenbankmigration (RDBMS) identifiziert: Schema-, Daten-, Stored Procedures-, Trigger- und View-Migration. Auch wenn Laszewski et al. in [17] speziell auf Oracle Produkte eingehen, sind die gezeigten Vorgehensweisen in verallgemeinerter Form für die meisten RDBMS gültig. In Kapitel Drei wird eine ausführliche Methodik zur Datenbankmigration, wie in [63] gezeigt, vorgeschlagen und in den darauf folgenden Kapiteln ausführlich darauf eingegangen. Kapitel Sieben beschreibt die Auswirkungen auf Schichten oberhalb des DBL. Die Vorgehensweisen von Laszewski et al. sind sehr umfassend und dienen daher als Grundlage für die hier vorgestellte gesamtheitliche Migrationsmethodik. Sie werden in dieser Arbeit in verallgemeinerter Form übernommen und um NoSQL und BLOB Datenspeicher sowie CDNs erweitert.

Informatica Cloud [64] ist ein SaaS Anbieter der u.a. eine sichere Datenmigration, -replikation und -synchronisation in die Cloud anbietet. Hierzu erlaubt eine Weboberfläche die Konfiguration *(„mappings, application connection information, and transformation rules“*) eines Synchronisations- oder Migrationsdienstes. Die Datenübertragung an sich wird von „Secure Agents“ übernommen, die lokal installiert werden und die Daten direkt zum Migrationsziel verschlüsselt übertragen. Bei einer Cloud-zu-Cloud Migration läuft der Migrations-Agent in einer virtuellen Umgebung außerhalb des lokalen Rechenzentrums. Die Ansätze von Informatica Cloud werden in das Cloud Datenmigrations-Tool einfließen oder der gesamte Dienst von Informatica Cloud kann ggf. komplett in das Tool eingebunden werden um doppelte Arbeit zu vermeiden. [65]

Das EU Projekt VISION Cloud [65] beschäftigt sich mit dem Aufbau eines inhaltszentrierten und verteilten Cloud Datenspeicherdienstes. Dieser soll u.a. interoperabel sein und eine einfache Portierbarkeit ohne Anbieter Lock-in, sowie einfache Migration von Daten ermöglichen. Die Arbeiten an „WP 3.2 Cloud Federation and Interoperability“, die für

diese Arbeit relevant sind und Überschneidungen beinhalten, sind allerdings erst für dieses Jahr (2012) geplant und noch nicht veröffentlicht [66].

2.3 Entscheidungsunterstützungssysteme

Da die in dieser Arbeit entwickelte Methodik u.a. bei der Entscheidungsfindung unterstützen soll, wie z.B. die lokalen Daten am besten in die Cloud migriert werden können, wird ein System zur Entscheidungsfindung (Decision Support System, DSS) benötigt.

Das hier zu entwickelnde DSS soll es dem Nutzer ermöglichen, interaktiv, zusammen mit dem System, sowohl Migrationsszenarien mit deren Ausprägungen zu bestimmen, als auch letztendlich passende Anbieter und benötigte Cloud Data Patterns zu identifizieren. Dabei handelt es sich nach Haettenschwiler [67] um ein kooperatives DSS und nach Power [68] um ein knowledge-driven, Desktop DSS. Grundsätzlich besteht ein DSS aus einer Datenbank mit der Wissensbasis, einem Modell mit Regeln und einer Benutzerschnittstelle [67], [68].

Zwei verwandte Arbeiten stellt Khajeh-Hosseini et al. in [69] vor. Es handelt sich hierbei um zwei Werkzeuge zur Entscheidungsunterstützung für die Migration von Anwendungen in das IaaS Cloud Deployment Modell. Das erste Tool ermöglicht eine Kostenvorhersage für verschiedene Anbieter, basierend auf einem UML Cloud Deployment Model mit annotierten Wachstum-Patterns. Das zweite Tool zeigt mögliche Vorteile und Risiken bei der Cloud Migration auf. Beide Tools sind jedoch nicht öffentlich zugänglich und somit im Rahmen dieser Arbeit nicht erweiterbar.

Menzel et al. [13] haben mit CloudGenius ein DSS für die Wahl eines IaaS Cloud Anbieters inklusive VM Image Auswahl mit Fokus auf die Migration von Webservern in die Cloud entwickelt. CloudGenius basiert auf dem Analytic Hierarchy Process (AHP), welcher sich hier nicht anbietet, da er sehr aufwändig und anspruchsvoll ist und vom Nutzer verlangt, dass er erst einmal alle in Frage kommenden Cloud Data Stores untersucht und beschreibt, sowie einen Kriterienkatalog erst erstellt und die Kriterien gegenseitig gewichtet. Das hier entwickelte Cloud Data Migration Tool soll einen vorgefertigten, aber erweiterbaren Katalog an Szenarien, Kriterien, möglichen Ausprägungen je Kriterium und Lösungen schon beinhalten. Diese ließen sich zwar zum Teil in die Implementierung von CloudGenius [70] bzw. dessen darunterliegenden Framework $(MC^2)^2$ [71] einpflegen, jedoch ist die prinzipielle Nutzerführung des geplanten Cloud Data Migration Tool gravierend anders und somit scheint der Nutzen der Wiederverwendung zu gering.

Nach einer Internetrecherche wurde kein passendes, frei verfügbares Framework gefunden, welches für ein kooperatives, knowledge-driven Desktop DSS genutzt werden kann. JBoss Drools [72], dass z.B. in einem medizinischen DSS eingesetzt wird [73], [74], würde zwar die Bedingungen erfüllen, allerdings würde der Einsatz dieser Anwendung die Komplexität unnötig erhöhen, da sich die hier benötigten Entscheidungen leicht algorithmisch lösen lassen und Drools bevorzugt für komplexere Aufgabenstellungen geeignet ist [75]. Der Framework Ansatz von Sprague [76], der ein DSS aus einer Modellbeschreibung und Wissensdatenbank generiert scheint ebenfalls zu komplex für das hier benötigte Cloud Data Migration Tool.

Das zu entwickelnde DSS soll sich allerdings an bestehende DSS angleichen und allgemeine DSS Konzepte benutzen.

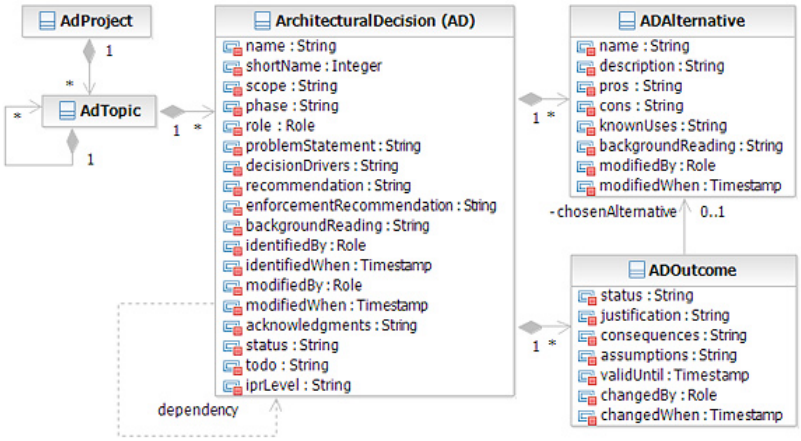

Abbildung 14 UML Meta-Modell zur Beschreibung von Architekturentscheidungen [77]

Zimmerman et al. beschreibt in [77] ein UML Meta-Modell zur Beschreibung von Architekturentscheidungen, wie in Abbildung 14 gezeigt. Dieses kann in der Implementierung später aufgegriffen und auf Migrationsszenarien und Lösungen von Cloud Anbietern erweitert werden. Auch die vier Ebenen des Entscheidungsprozesses: *„Business Execution Level, Conceptual Level, Technology Level and Vendor Asset Level“*, wie zum Teil in [78] und Abbildung 15 von Zimmerman gezeigt, werden übertragen auf das Cloud Data Migration Tool. Auf konzeptioneller Ebene steht die Auswahl des Migrationsszenarios, auf technologischer Ebene die Cloud Data Hosting Solution sowie Data Layer Patterns und auf der Herstellerebene der Cloud Data Store.

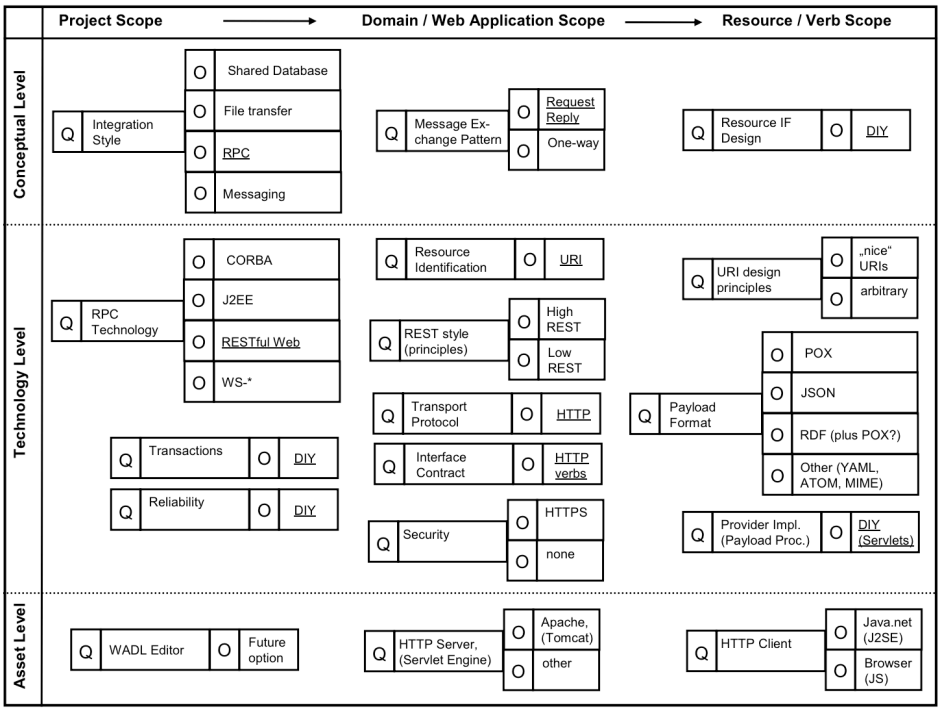

Abbildung 15 Beispiel einer Entscheidungshierarchie [79]

2.4 SQL versus NoSQL

Neben Datenspeichern für Binary Large Objects (BLOBs) existieren heute vor allem RDBMS, die mit SQL angesprochen werden [80], und NoSQL Datenbanken [81], [82], die z.B. aus einem schemalosen Schlüssel-Attribute Speicher (Key-Value(s) Store) bestehen und mit wenigen primitiven Methoden wie put, query oder delete angesprochen werden.

Ein Großteil der Softwaresysteme im Enterprise Umfeld verwenden RDBMS um ihre Daten vorzuhalten [80]. Die Daten und Beschreibungen der Daten aus solch einem RDBMS müssen bei der Cloud Datenmigration in einen geeigneten Cloud Datendienst importiert werden. Dabei stellt sich prinzipiell die Frage, ob der Cloud Datendienst auch zwangsläufig ein RDBMS sein muss, oder ob auch ein besser skalierbarer NoSQL Dienst verwendet werden kann. Wann ist also eine Migration von SQL nach NoSQL möglich?

Prinzipiell sind RDBMS sehr gut für transaktionsorientierte Anwendungen wie „Systeme, die Bestelleingänge, Flugticketreservierungen, Lohn- und Gehaltsabrechnungen, Mitarbeiterdatensätze, Fertigungs- und Versanddaten verwalten“ [8] geeignet. Bei diesen Systemen werden häufig in einem Schritt (Transaktion) Werte eines Datensatzes verringert und Werte eines anderen Datensatzes erhöht oder andere Änderungen an mehreren Tabellen gleichzeitig vorgenommen. RDBMS stellen diese Transaktionen mit den sogenannten ACID-Kriterien (Atomicity, Consistency, Isolation, Durability) sicher. Damit wird jede Transaktion atomar, also entweder ganz oder gar nicht ausgeführt, führt von einem konsistenten Zustand zum nächsten, läuft isoliert von anderen Transaktionen ab und die Ergebnisse sind persistent gespeichert.

Hingegen sind verteilte NoSQL Datenbanken auf hohe Verfügbarkeit und sehr geringe Latenz ausgelegt, was in der Regel durch Einschränkungen in der Komplexität von Anfragen, transaktionalem Verhalten oder der strikten Konsistenz ermöglicht wird [80].

Das CAP Theorem [83] zeigt z.B., dass bei verteilten Datenbanken bei einem Netzwerkproblem nicht gleichzeitig vollständige Konsistenz und Verfügbarkeit sichergestellt werden kann. Aber auch ohne Netzwerkprobleme muss bei verteilten Datenbanken ein Kompromiss zwischen geringer Latenz und strikter Konsistenz gefunden werden. All diese Abwägungen beschreibt PACELC:

> *[If] there is a partition (P), how does the system trade off availability and consistency (A and C); else (E), when the system is running normally in the absence of partitions, how does the system trade off latency (L) and consistency (C)?* [84]

Eine Abbildung von SQL Anfragen im Kontext von Transaktionen auf Anfragen für NoSQL Datenbanken ist also nicht ohne Weiteres möglich. Je nach Anwendungskontext muss entschieden werden, ob die ACID-Kriterien benötigt werden oder ob darauf verzichtet werden kann um performantere und leichter skalierbare NoSQL Datenspeicher zu verwenden. Pritchett beschreibt in [85] Möglichkeiten wie man ACID-Kriterien sinnvoll aufweichen kann um NoSQL Datenbanken zu verwenden, auch wenn dies nicht für alle Transaktionen möglich oder sinnvoll ist.

Es existieren allerdings auch Produkte, wie der NoSQL Datenspeicher in Googles App Engine, die Transaktionen auf Entitätsgruppen-Ebene (ähnlich einer Transaktion mit Zugriffen innerhalb einer einzelnen Tabelle) oder über bis zu fünf Entitätsgruppen hinweg (XG Transactions) unterstützen. Eine Entitätsgruppe ist vergleichbar mit sehr stark zusammenhängenden Tabellen in der relationalen Welt (denormalisierte Kind-Beziehungen). Aber auch hier kann es laut Google in extrem seltenen Fällen vorkommen, dass eine solche Transaktion über Entitätsgruppen committed wird, obwohl dem Programm angezeigt wird, dass sie fehlgeschlagen sei. Daher wird empfohlen bei der Verwendung von Transaktionen in Google NoSQL Datenspeicher, diese idem potent zu gestalten. [86]

Andere NoSQL Datenspeicher wie MongoDB [87] oder Cassandra [88] unterstützen keine Transaktionen über Entitätsgruppen hinweg. Sie ermöglichen aber Atomarität auf Objektebene, das heißt wenn mehrere Änderungen an einem Objekt zugleich vorgenommen werden, werden entweder alle oder keine Änderungen persistiert.

Nach Einschätzung des Autors werden sich in Zukunft RDBMS und NoSQL Datenspeicher aufeinander zubewegen und mehr Spielräume bieten um genau zum Anwendungskontext zu passen. Zu sehen ist dies zum Beispiel schon beim AWS Dienst DynamoDB, welcher Lesezugriffe sowohl mit *Strong Consistency*, als auch mit schnellerer und günstigerer *Eventual Consistency* unterstützt [89]. Weiter werden verteilte RDBMS leichter administrierbar werden und über (bessere) Auto-Sharding Funktionen verfügen um ihre Daten selbständig in Clustern zu verteilen [90].

3 Grundlagen

Im folgenden Kapitel werden Grundlagen beschrieben, auf die später in dieser Arbeit aufgebaut wird.

3.1 Datenschicht im 3-Schichten Modell

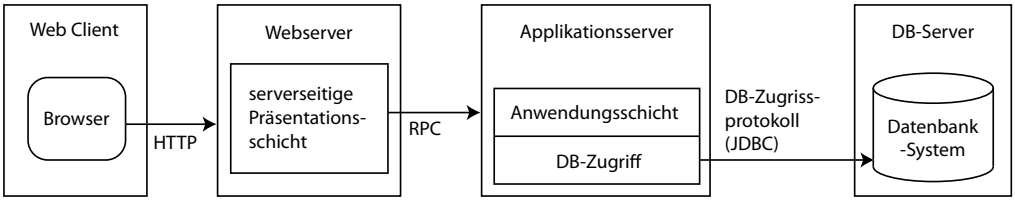

Abbildung 14 Typische N-Schichten-Architektur [5]

Eine typische N-Schichten-Architektur ist in Abbildung 14 zu sehen. Hier wird die typische 3-Schichten-Architektur, bestehend aus Präsentationssicht, Anwendungssicht und Datenschicht noch weiter unterteilt. Die Präsentationssicht setzt sich aus einem Browser-Client und einer serverseitigen Präsentationsschicht zusammen und die Datenschicht aus einem DAL (DB-Zugriff) und DBL (Datenbanksystem).

Genau diese Schichten-Architektur ist im Cloud Computing Umfeld häufig anzutreffen. Ein gehostetes RDBMS oder ein DaaS Dienst fungiert als DBL, ein Anwendungscontainer mitsamt der Geschäftsanwendung läuft in einem Infrastructure-as-a-Service (IaaS, wie z.B. Amazon EC2) oder Platform-as-a-Service (PaaS, wie z.B. Google App Engine) Deployment und generiert dynamische Webseiten für einen Browser-Client. Dabei können der Webserver und Anwendungscontainer eng miteinander verbunden sein und auf dem gleichen Knoten laufen oder getrennt sein, wenn nicht nur dynamisch generierte Inhalte zum Browser-Client ausgeliefert werden.

Beim Aufteilen einer Anwendung in verschiedene Schichten werden auch immer Abstraktionsebenen eingeführt, welche eine kleine Latenz mit sich bringen. Werden zusätzlich die verschiedenen Schichten auch noch auf verschiedene (virtuelle) Maschinen verteilt und müssen über ein Netzwerk miteinander kommunizieren, kommt noch eine erhebliche Latenz durch den Transport von Daten über das Netzwerk hinzu.

Um diese Latenzen möglichst gering zu halten, wird versucht die Berechnungen räumlich möglichst nah an die Daten zu bringen [91], um hier die Latenz durch wenige „Hops“ zu minimieren. Dieser Ansatz wird *function-shipping* [92] genannt. Beim alternativen *data-shipping* Ansatz werden die Daten zur Berechnung gebracht, was bei großen Datenmengen in der Regel langsamer ist als function-shipping. Ein Mittelweg geht zum Beispiel AWS mit Elastic MapReduce (EMR) und DynamoDB bzw. S3 [93]. Dieser ermöglicht es komplexe SQL-Abfragen auf der NoSQL Datenbank DynamoDB oder dem BLOB Datenspeicher S3 auszuführen. Dabei ist EMR räumlich nah am DynamoDB und S3 Dienst angesiedelt und hat somit kurze Latenzzeiten für das benötigte data-shipping. Da DynamoDB und S3 selbst keine komplexen Filter oder Joins erlauben, müssen zuerst die zu analysierenden Daten in EMR geladen werden. Danach können die auszuführenden Analysen in Form von SQL-Befehlen über ssh direkt an EMR geschickt und die Analyseergebnisse in DynamoDB oder S3 gespeichert werden. Damit müssen die Daten zur Analyse nicht erst weit transportiert werden, sondern werden nah an der Quelle analysiert.

3.2 Cloud Computing

Laut NIST Definition [10] ist Cloud Computing

> *"... a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models."*

Cloud Computing zeichnet sich durch Charakteristiken wie *On-demand self-service, Broad network access, Resource pooling, Rapid elasticity* und *Measured Service* aus. Es bietet als Deployment-Optionen *Private Cloud, Community Cloud, Public Cloud* und *Hybrid Cloud*. Diese können in Form von *Software as a Service (SaaS), Platform as a Service (PaaS)* und *Infrastructure as a Service (IaaS)* genutzt werden. [10], [62]

In dieser Arbeit spielen vor allem die Service Modelle IaaS und PaaS eine Rolle. Beim IaaS Modell hat der Benutzer die größten Freiräume bezüglich der installierbaren Software, muss sich aber selbst und die Administration, den Betrieb und die Wartung dieser Software kümmern. Beim PaaS Modell hingegen stellt der Cloud Anbieter verschiedene Dienste, wie z.B. einen hochskalierbaren Datenbankdienst der komplett gemanagt ist, zur Verfügung. Damit sind die Freiheiten in der Konfiguration und Auswahl der Software zwar eingeschränkt, dafür besteht aber auch kein Administrations-, Betriebs- oder Wartungsaufwand. [10]

Als Deployment Optionen kommen in dieser Arbeit alle Möglichkeiten vor. Für persönliche oder unternehmenskritische Daten werden häufig Private, Community oder Hybrid Clouds bevorzugt, ansonsten bieten Public Clouds den Vorteil, dass sie nicht selbst gemanagt werden müssen. [10]

3.3 Amazon Web Services

Amazon Web Services (AWS) bot 2006 die ersten Infrastrukturdienste als Web Services an, die später als Cloud Computing bekannt wurden. Heutzutage bietet AWS „eine hoch verfügbare, skalierbare und kostengünstige Rechenplattform in der Cloud, die in mehreren Hunderttausend Unternehmen in 190 Ländern weltweit eingesetzt wird". [94]

Die in dieser Arbeit direkt referenzierten Dienste werden im Folgenden kurz beschrieben: [95]

- Amazon Elastic Compute Cloud (EC2): virtuelle Rechner, auf denen u.a. ein Datenbankserver betrieben werden kann (IaaS)
- Amazon Elastic MapReduce: Dienst zur Analyse von großen Datenmengen (PaaS)
- Amazon SimpleDB: nicht-relationale DaaS Lösung mit SQL ähnlicher Abfragesprache (PaaS)
- Amazon DynamoDB: nicht-relationale DaaS Lösung mit definiertem Datendurchsatz und vorhersagbarer Antwortzeit (PaaS)
- Amazon Relational Database Service (RDS): gehostete und administrierte Version von RDBMS wie MySQL, SQL Server und Oracle 11g (PaaS)
- Amazon Simple Storage Service (S3): redundanter Datenspeicher (PaaS)

- Amazon Elastic Block Store (EBS): in virtuelle Rechner (EC2) einbindbarer persistenter Speicherbereich (unformatiert, Zugriff auf Blockebene) auf dem z.B. ein Datenbankserver seine Daten ablegen kann (IaaS)
- Amazon Virtual Private Cloud (VPC): „Private Cloud" Lösung in Amazons Cloud Rechenzentrum (IaaS)
- Amazon CloudFront: CDN mit starker Amazon S3 Integration (PaaS)
- Amazon Import/Export: Dienst zum Import oder Export von großen Datenmengen mittels tragbarer Speichergeräte oder auch zum Im- und Exportieren von virtuellen Maschinenabbildern über das Internet

Da auf EC2, SimpleDB und RDS in dieser Arbeit und dem Cloud Data Migration Tool näher eingegangen wird, werden diese Dienste im Folgenden detaillierter beschrieben:

Amazon EC2 bietet die Möglichkeit unbegrenzt viele Instanzen einer virtuellen Maschine parallel laufen zu lassen. Dabei können diese zustandsbehaftet oder zustandslos sein, d.h. entweder nach einem Neustart ihre alten Informationen, wie gespeicherten Daten, noch besitzen oder bei jedem Neustart alle gespeicherten Daten vergessen. In jede EC2 Instanz lassen sich EBS Volumes einbinden, welche ihre Daten persistent speichern. Somit lassen sich Datenbankserver auch performant mit zustandslosen EC2 Instanzen betreiben, indem die Datenbanken selbst persistent auf einem EBS Volume liegen. Da es sich in diesem Fall um eine IaaS Lösung handelt, müssen auch sämtliche Betriebssystem und Datenbankserver Updates selbst installiert werden um stets aktuellen Sicherheitsstandards zu genügen. [96]

Amazon SimpleDB ist ein begrenzt skalierbarer NoSQL Datenspeicherdienst. Durch die maximale Attributgröße von 1KB und maximale Domaingröße von 10 GB (eine Domain entspricht grob einer Tabelle) lässt sich der Dienst nicht für jedes Szenario einsetzen. Sollen zum Beispiel größere Texte gespeichert werden, können diese nicht im NoSQL Datenspeicher direkt hinterlegt werden, sondern müssen z.B. auf Amazon S3 gespeichert werden und SimpleDB enthält nur eine Referenz auf den gespeicherten Text innerhalb von S3. SimpleDB bietet eine SQL-ähnliche Abfragesprache zum Filtern und Sortieren von Ergebnismengen, dabei sind keine Domainübergreifenden Joins erlaubt, die Ermittlung der Ergebnisse darf nicht länger als fünf Sekunden benötigen und die Größe der Ergebnismenge darf 1 MB nicht überschreiten. [97], [98]

Amazon RDS ist ein Cloud Hosting Angebot von verschiedenen relationalen Datenbankprodukten wie MySQL, SQL Server und Oracle 11g. Die angebotenen Funktionen fallen je nach ausgewähltem Produkt unterschiedlich aus. Der RDS Dienst mit MySQL als Datenbankprodukt unterstützt z.B. manuelle Snapshots, Read-Replicas, Backups, Point-in-time Recovery, Multi-Availability-Zone Deployment, automatisches Failover und automatische Produktupdates. Die anderen beiden Varianten beinhalten z.B. keine Read-Replicas und Multi-Availability-Zone Deployment. [99]

3.4 Microsoft Azure

Windows Azure startete 2008 [100] und bietet mittlerweile eine Vielzahl von Diensten an, wobei die in dieser Arbeit referenzierten Dienste im Folgenden kurz beschrieben werden: [101], [102], [103]

- Virtual Machine (VM) Role: virtueller Windows Server 2008 RC 2 Rechner (IaaS)
- SQL Database (früher: SQL Azure): eine gehostete und administrierte Variante des SQL Server speziell für die Cloud (PaaS)

- Tables: nicht-relationale DaaS Lösung (PaaS)
- Blobs: Speicherdienst für unstrukturierte binäre Daten (PaaS)
- Windows Azure Drives: in virtuelle Rechner (VM Role) einbindbarer persistenter Speicherbereich (NTFS) (PaaS)
- SQL Data Sync: ein Dienst zur asynchronen Synchronisation von lokalen SQL Server und Azure SQL Database Datenbanken

Da auf SQL Database in dieser Arbeit und dem Cloud Data Migration Tool näher eingegangen wird, wird dieser Dienst im Folgenden detaillierter beschrieben:

SQL Database ist eine für den Cloud Betrieb optimierte Version des SQL Servers. Der Dienst wird von Microsoft in der Azure Cloud betrieben und gewartet und ist weitestgehend mit den SQL Funktionen des SQL Server kompatibel. Die maximale Datenbankgröße ist allerdings auf 150 GB beschränkt. Zusätzlich gibt es die Möglichkeit eine Synchronisation zwischen SQL Database Instanzen oder einer SQL Database Instanz und einer Datenbank eines lokalen SQL Servers einzurichten. Allerdings kann diese Synchronisation maximal alle fünf Minuten getriggert werden.

3.5 Google

Google begann 2008 eine PaaS Umgebung aufzubauen und mit der Zeit verschiedene Datenspeicher einzubinden, welche im Folgenden kurz beschrieben werden: [104]

- Google App Engine Datastore: nicht-relationale DaaS Lösung, u.a. mit Unterstützung von ORMs wie JDO und JPA
- Google Cloud SQL: eine in der Cloud gehostete MySQL Variante
- Google Cloud Storage: ein Amazon S3 Protokoll-kompatibler Cloud Dateispeicher

Da auf den Google App Engine Datastore und Google Cloud SQL Dienst in dieser Arbeit und dem Cloud Data Migration Tool näher eingegangen wird, werden diese Dienste im Folgenden detaillierter beschrieben:

Der Google App Engine Datastore in seiner aktuellen Version (High Replication Datastore) ist ein schemaloser, eventual consistent NoSQL Dienst, welcher über eine JDO und JPA Schnittstelle, als auch über eine Low-level API ansprechbar ist. Operationen wie Joins, Filter auf Ungleichheit oder Unterabfragen (Subquerys) werden nicht unterstützt. Ein Datensatz darf max. 1 MB groß sein und mehr als 20.000 Indexeinträge pro Entität bzw. mehr als 200 Indexe insgesamt sind nicht erlaubt. Dafür skaliert der Dienst sehr gut auch bei großen Datenmengen.

Alternativ zum Amazons RDS Dienst mit MySQL Engine bietet auch Google mit dem Cloud SQL Dienst eine in der Cloud gehostete, wartungsfreie MySQL Instanz an. Zwar lassen sich hier verschiedene Versionen der Open Source Datenbank auswählen, allerdings ist es nicht möglich die Konfigurationen anzupassen, wie es bei RDS der Fall ist.

4 Gesamtheitliche Cloud Datenmigration-Methodik

Dieses Kapitel bildet den Schwerpunkt dieser Arbeit. Es wird ausgehend von identifizierten Migrationsszenarien eine Taxonomie zur Cloud Datenmigration erstellt und die Migrationsszenarien werden auf Cloud Data Hosting Lösungen und Cloud Data Patterns abgebildet. Anschließend werden die verschieden Arten von Daten, die bei der Datenmigration eine Rolle spielen, analysiert und Auswirkungen der Cloud Datenmigration auf höhere Anwendungsschichten dargelegt. Schließlich werden die zuvor erschlossenen Informationen in eine ganzheitliche Cloud Datenmigrationsmethodik eingearbeitet und Fragebögen entwickelt, welche bei der Analyse und Umsetzung von Cloud Datenmigrationsprojekten unterstützen.

In den folgenden Abschnitten 4.1 und 4.2 wird der Begriff Cloud allgemein verwendet und umfasst sämtliche Deployment und Service Modelle (siehe Abschnitt 3.2). Erst in Abschnitt 4.3 werden die Auswirkungen der verschiedenen Deployment und Service Modelle für die einzelnen Migrationsszenarien differenziert betrachtet.

4.1 Migrationsszenarien

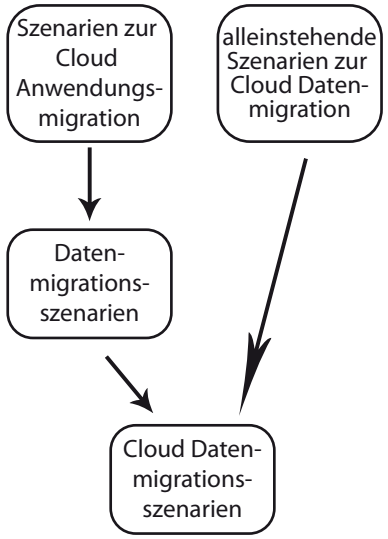

Abbildung 15 Vorgehensweise zur Identifikation von Cloud Datenmigrationsszenarien

Um Cloud Datenmigrationsszenarien zu identifizieren, werden, wie in Abbildung 15 gezeigt, im Folgenden zuerst allgemeine Szenarien zur Cloud Anwendungsmigration identifiziert und entsprechende Datenmigrationsszenarien daraus abgeleitet und danach alleinstehende Cloud Datenmigrationsszenarien identifiziert.

4.1.1 Szenarien zur Cloud Anwendungsmigration

Von den typischen Anwendungsarten für die Cloud, die in den Abschnitten 2.1 und 2.2 basierend auf der erfolgten Literaturrecherche beschriebenen wurden, qualifizieren sich zusammengefasst die folgenden Anwendungsarten als Kandidaten für eine Migration in die Cloud:

- Webanwendung [23]
  - Hochskalierbare Webseiten [33], [32]
  - Enterprise Anwendungen [28]
  - Kundenorientierte Anwendungen und Mobile Apps [27], [29], [30]
  - Online Spiele [17]
- Legacy Anwendungen [24]
- Parallele Stapelverarbeitung [26], [25] , [29]

- Hintergrundverarbeitung [62]
    - Simulation
    - Berechnungen für Desktop Anwendungen und Mobile Apps [31], [62]
- Analyse auf großen Datenmengen [105]
    - Business Intelligence [62]

Soll nun eine der identifizierten Anwendungstypen komplett in die Cloud migriert werden, muss auch die Datenschicht migriert werden, wodurch sich aus den folgenden Anwendungsmigrationsszenarien die in *kursiv* hervorgehobenen Datenmigrationsszenarien ableiten lassen:

4.1.1.1 Webanwendungen

Hochskalierbare Webseiten

Hochskalierbare Webseiten, wie Webshops [106], Blogs oder Webseiten zu Veranstaltungen [107], sind häufig nicht statisch, sondern besitzen dynamischen Inhalt und werden bei jedem Aufruf neu zusammengestellt oder können aus einem Cache geladen werden, sofern sich kürzlich nichts an den dynamischen Inhalten verändert hat.

Die Präsentations- und Anwendungsschicht lässt sich durch Ortstransparenz und Zustandslosigkeit horizontal skalierbar machen [105]. Um der Hochskalierbarkeit der gesamten Anwendung aber gerecht zu werden muss auch die Datenschicht hochskalierbar sein, was häufig aufwändiger ist, sofern sie auf einem RDBMS beruht.

Neben dem einfachen *Outsourcing* [108] der Webseite inklusive Datenschicht kann zusätzlich die Skalierbarkeit, Verfügbarkeit und Performance der Datenschicht [105], [108] durch *Datenverteilung (Sharding)* [109], *geographische Replikation* [108] und Nutzung von *hochskalierbaren Datenspeichern* [110] (NoSQL) verbessert werden. Bei statischen Webseiten, die hochskalierbar und weltweit schnell erreichbar sein sollen, ist die Cloud Migration am einfachsten, da die Inhalte nur über ein *Content Delivery Network* (CDN) [111] weltweit verteilt werden müssen.

Enterprise Anwendungen

Enterprise Anwendungen, wie ERP-, Finance- [107], Human Resource- (HR) [9] oder CRM-Systeme, die im eigenen oder externen Rechenzentrum virtualisiert gehostet werden oder von einem Application Service Provider (ASP) angeboten werden, können in der Regel ohne großen Aufwand in eine IaaS Cloud migriert werden [17]. Die dadurch entstandenen Vorteile sind aber häufig gering, da das alleinige Portieren von Enterprise Anwendungen in die Cloud zwar ggf. eine höhere Verfügbarkeit und geringere Betriebskosten, aber nicht automatisch eine bessere Skalierbarkeit mit sich bringt. Um Elastizität in Enterprise Anwendungen hineinzubringen, müssen sie überwiegend für die Cloud angepasst werden und spezielle elastische und hochskalierbare Cloud Dienste benutzen [17]. Hierbei spielt auch die Anpassung der Datenschicht eine Rolle, wo die gleichen Herausforderungen wie bei hochskalierbaren Webseiten auftreten, welche durch *Outsourcing, Datenverteilung, geographische Replikation, hochskalierbare Datenspeicher* und ein *CDN* gelöst werden können.

Prinzipiell stellt sich allerdings die Frage ob eine Enterprise Anwendung überhaupt elastisch und hochskalierbar sein muss, da die Nutzung in der Regel nicht stark variiert (außer bei Saisongeschäften) und auch nicht schnell anwächst, da sie sich am Wachstum von Unternehmen und Märkten orientiert, welche in der Regel organisch wachsen. An-

ders sieht es aus, wenn eine Enterprise Anwendung mit der Migration in die Cloud nun von mehreren Kunden gleichzeitig und isoliert voneinander genutzt werden soll und somit mandantenfähig sein muss.

Bei internationalen Unternehmen mit mehreren Standorten, kann, bezogen auf die Datenschicht einer Enterprise Anwendung zusätzlich noch die Verteilung der Daten nah an die Unternehmensstandorte sinnvoll sein (*Datenverteilung*) um niedrige Latenzen zu ermöglichen. Dies funktioniert allerdings nur, wenn diese Verteilung auf natürlichen geographischen Gründen beruht, wie z.B. wenn ERP oder CRM Daten von einem Standort selten in anderen Standorten benötigt werden.

Kundenorientierte Anwendungen
Kundenorientierte Anwendungen, wie soziale Netzwerke [17], und Online Spiele [112] verhalten sich ähnlich wie hochskalierbare dynamische Webseiten. Sie zeigen auch dynamische (Web)-Inhalte an und müssen z.B. je nach Tageszeit oder Hypephase elastisch auf die Änderung der Anfragenrate reagieren. Auch native Mobile Apps [113], die lediglich Daten für mobile Endgeräte entsprechend aufbereitet anzeigen, hybride Mobile Apps oder mobile Webanwendungen können mit kundenorientierten Anwendungen verglichen werden. Sie laufen lediglich auf einem Ressourcen-ärmeren Endgerät. Damit können Kundenorientierte Anwendungen und die hier beschriebenen Mobilen Apps bezüglich der Cloud Datenmigration die gleichen Strategien wie hochskalierbare dynamische Webseiten verwenden (*Outsourcing, Datenverteilung, geographische Replikation, hochskalierbare Datenspeicher* und ein *CDN*) einsetzen.

4.1.1.2 Legacy Anwendungen
Legacy Anwendungen, also große Altsysteme die noch verwendet werden, aber nicht mehr produktiv gewartet werden können, sind nur mit größerem Aufwand in die Cloud migrierbar. Laszewski führt in [114] die folgenden möglichen Vorgehensweisen an: Neuschreiben bzw. Umstrukturieren, Plattformwechsel, Tool-gestützte automatisierte Konvertierung, Emulation durch SOA, Web Services oder Screen Scraper und Datenmodernisierung.

Elastizität in eine Legacy Anwendung hineinzubringen wird häufig auf Grund der fehlenden Dokumentation und des fehlenden Know-hows schwer möglich sein. Somit ist häufig der Betrieb einer Legacy Anwendung in der Cloud in Verbindung mit Konnektoren und Adaptern (SOA) die einzige Alternative um die Funktionen der Anwendung anderen Anwendungen zur Verfügung zu stellen. Die Migration der Datenschicht wird sich hier in der Regel auf die einfache Portierung (*Outsourcing*) der Datenbank in die Cloud beschränken. [113]

4.1.1.3 Parallele Stapelverarbeitung
Parallele Stapelverarbeitungen, wie Bildverarbeitung [115], Datenkonvertierung [116] oder Berichtserstellung nach Geschäftsschluss (End of Day), haben in der Regel keine graphische Benutzungsschnittstelle (GUI) sondern bearbeiten große Datenmengen skriptbasiert. Diese Verarbeitung in die Cloud zu migrieren ist sinnvoll, wenn sich die Aufgaben parallelisieren lassen und man somit für die gleichen Kosten schneller zu Ergebnissen kommen kann. Dies ist möglich, da in der Cloud nur für die genutzte Rechenleistung bezahlt wird und somit sechs Rechner á einer Stunde genauso teuer sind wie einen Rechner sechs Stunden zu betreiben. Wenn die Daten bisher im lokalen Rechenzentrum verarbeitet wurden, müssen sie zuerst in die Cloud migriert (*Outsourcing*) und die Ergebnisse nach der Verarbeitung wieder zurück ins lokale Rechenzentrum

transferiert werden. Neben dieser Transportanforderung an die Datenschicht müssen die Daten auch schnell genug den parallel ausgeführten Skripten zur Verfügung stehen, was möglicherweise die Verwendung eines spezifischen Cloud Datenspeichers (*hochskalierbarer Datenspeicher*) bedingt.

4.1.1.4 Hintergrundverarbeitung

Die Hintergrundverarbeitung hat im Gegensatz zur parallelen Stapelverarbeitung das Ziel die Ergebnisse mit möglichst kurzer Latenz an einen Client zurückzuliefern. Der Client selbst verfügt nicht über genügend Ressourcen um bestimmte Aufgaben selbst auszuführen und nutzt daher Dienste einer Serveranwendung.

Simulation

Simulation, wie Vorhersagen anhand wissenschaftlicher Modelle [117], sind ähnlich zur parallelen Stapelverarbeitung. Allerdings sind sie teilweise nicht so stark parallelisierbar und werden in der Regel auch nicht regelmäßig wiederholt ausgeführt. Um dennoch schnelle Ergebnisse zu bekommen, werden häufig Ansätze des HPC verwendet. Im Vergleich zu Business Intelligence müssen für Simulationen nicht zwangsläufig große Datenmengen betrachtet werden, sondern es können auch komplexe Algorithmen auf kleinere Datenmengen angewendet werden. Die Daten die zur Simulation benötigt werden, müssen zuerst in die Cloud migriert (*Outsourcing*) und die Ergebnisse der Simulation anschließend in das lokale Rechenzentrum zurück kopiert werden. Möglicherweise müssen die Datenmengen auch besonders schnell und in großer Anzahl den Berechnungen zur Verfügung stehen oder (Zwischen)-Ergebnisse schnell persistiert werden können um keinen Flaschenhals darzustellen. Dazu können spezifische Cloud Datenspeicher (*hochskalierbarer Datenspeicher*) verwendet werden.

Berechnungen für Desktop Anwendungen

Berechnungen für Desktop Anwendungen, die z.B. 3D-Szenen rendern oder ressourcenintensive Berechnungen durchführen, und Mobile Apps, die z.B. Zugriff auf entfernte Daten geben wie Dropbox [118] und Spotify [119], Zugriffe auf große Datenbanken bieten wie Foodspotting [118] oder komplexe Berechnungen outsourcen wie PicTranslator [119], kommunizieren in der Regel über eine Anwendungsschnittstelle (API) mit einer Serveranwendung. Die Präsentationssicht und kleinere Berechnungen werden also von der Desktop Anwendung oder mobilen App übernommen. Die Serveranwendung und dahinterliegende Datenschicht muss allerdings in den meisten Fällen hochskalierbar und elastisch sein um ähnlich wie bei den kundenorientierten Anwendungen zu bestimmten Tageszeiten oder in Phasen mit sehr hohem Lastaufkommen eine konstante schnelle Antwortzeit zu liefern. Somit können Berechnungen für Desktop Anwendungen und mobile Apps bezüglich der Cloud Migration von Anwendungs- und Datenschicht die gleichen Strategien wie hochskalierbare dynamische Webseiten verwenden: *Outsourcing, Datenverteilung, geographische Replikation, hochskalierbare Datenspeicher* und ein *CDN*.

4.1.1.5 Analyse auf großen Datenmengen

Business Intelligence (BI) [120], wie Business Analytics [121] oder Data Mining Anwendungen [122], [123], basieren in der Regel auf einem Data Warehouse (DW). Hier werden Daten speziell so für Online Analytical Processing (OLAP) präpariert, um darauf schnell Analysen durchführen zu können. Bei der Migration eines DW in die Cloud (*Outsourcing*) muss nicht nur dafür gesorgt werden, dass die bisherigen Daten migriert werden, sondern auch dass der ETL Prozess angepasst wird um die zu analysierenden Daten

der Enterprise Anwendung zukünftig in das DW in der Cloud zu kopieren [17] (*Arbeiten auf Datenkopie, Synchronisation*).

4.1.1.6 Zusammenfassung

Die eben gezeigten Szenarien zur Cloud Anwendungsmigration beinhalten bezogen auf die Datenmigration die Möglichkeiten des Outsourcings, der geographische Replikation, der Datenverteilung, die Verwendung von hochskalierbaren Datenspeichern, CDNs, Arbeiten auf Datenkopien und Einsatz von Synchronisationsmechanismen. Diese Möglichkeiten werden in den folgenden Abschnitten näher beschrieben und können auch kombiniert werden.

Anstatt Daten dauerhaft in die Cloud auszulagern (Outsourcing) besteht auch die Möglichkeit zusätzlich benötigte Ressourcen in Lastspitzen durch Cloud-Ressourcen auszugleichen, anstatt sie im eigenen Rechenzentrum vorzuhalten. Dieser Ansatz wird *Off-Loading of Peak Loads (Cloud Burst)* [10] genannt und kann in Zeiten von Lastspitzen entweder den gesamten DBL in die Cloud verschieben oder nur die zusätzlich benötigten Ressourcen aus der Cloud nutzen. Letzteres geschieht z.B. bei der partiellen Migration des DBL in die Cloud indem nur die Daten in die Cloud verschoben werden, welche von vielen Lese- und/oder Schreibzugriffen betroffen sind.

Neben diesen aus der Cloud Anwendungsmigration heraus entstehenden Möglichkeiten bei der Datenmigration existieren weitere Möglichkeiten bei alleiniger Betrachtung der Cloud Datenmigration. Im Folgenden werden zunächst die Möglichkeiten, die aus der Cloud Anwendungsmigration heraus entstanden sind beschrieben und anschließend weitere Möglichkeiten zur Datenmigration identifiziert, die auf der alleinigen Migration des DBL basieren (Backup, Archivierung, Datenimport aus der Cloud).

4.1.2 Reines Outsourcing des DBL

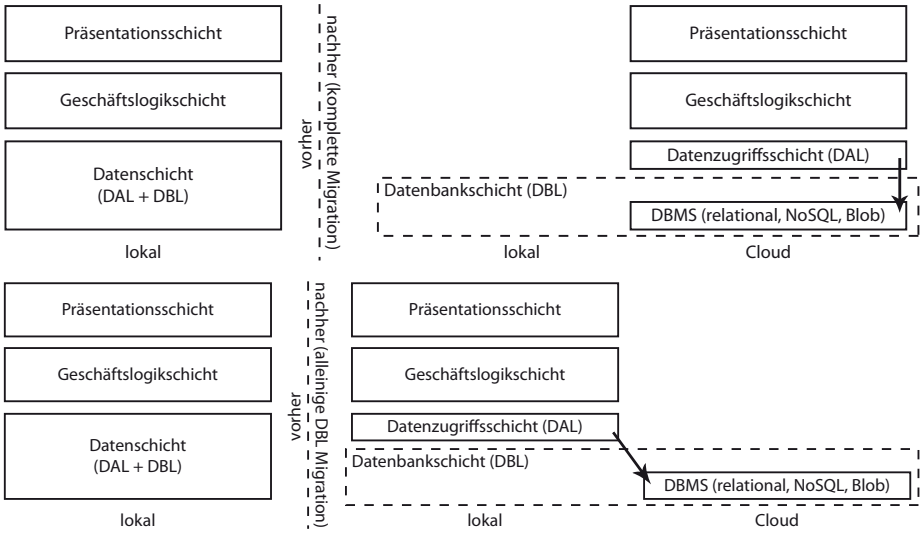

Abbildung 16 Vorher-Nachher-Vergleich beim reinen Outsourcing des DBL

Outsourcing des DBL bedeutet die Migration des lokalen DBL in die Cloud ohne dabei den Typ des Datenspeichers (RDBMS, NoSQL oder BLOB Datenspeicher) zu verändern, wie in Abbildung 16 zu sehen ist. Dabei kann der DBL in der Cloud selbst verwaltet sein oder vom Cloud-Anbieter.

Bei der *Selbstverwaltung* des DBL in der Cloud (IaaS) kann, sofern der zugehörige Datenbankserver nur für die zu migrierende Datenbank benutzt wird und in einer virtuali-

sierten Umgebung läuft, das *virtuelle Maschinen Abbild* in die Cloud direkt *importiert* werden [107]. Nach der Anpassung der Datenbankverbindungskennungen (Connection-Strings) und Firewall-Einstellungen können die lokalen oder ebenfalls in die Cloud migrierten höheren Anwendungsschichten den migrierten DBL ohne weitere Anpassungen verwenden. Es besteht auch die Möglichkeit *fertig konfigurierte Cloud Images* mit installierten Datenspeichern zu verwenden und die lokale Datenbank zu exportieren und in eine Instanz eines vorkonfigurierten Cloud Images zu importieren [96]. Statt eines fertig konfigurierten Cloud Images kann auch ein *neues Cloud Image* erzeugt werden [124].

Neben der Selbstverwaltung von Datenspeichern kann der lokale DBL auch in einen *gemanagten Cloud Datenspeicher* [96] (DaaS, z.B. als Dienst innerhalb eines PaaS) migriert werden. Hierbei müssen die Daten aus dem lokalen DBL exportiert und in den gemanagten Datenspeicher importiert werden. Neben den oben genannten Anpassungen der höheren Anwendungsschichten bei der Selbstverwaltung kann es bei dieser Variante des Outsourcings dazu kommen, dass bestimmte Funktionen oder Konfigurationen in den gemanagten Datenspeichern nicht zur Verfügung stehen. Dies kann weitere Auswirkungen auf höhere Anwendungsschichten haben, sofern diese nicht mehr vorhandenen Funktionen oder Konfigurationen von höheren Anwendungsschichten genutzt werden.

Ein Beispiel für die Migration eines RDBMS in die Cloud ist das Verschieben einer lokalen MySQL Datenbank in die Cloud. Hierbei kann der RDS Dienst in der MySQL Konfiguration von AWS genutzt, ein existierendes virtuelles Maschinen Abbild importiert oder ein vorkonfiguriertes OpenSolaris Cloud Image genutzt werden. [96]

Neben RDBMS können auch NoSQL Datenspeicher oder BLOB Datenspeicher in die Cloud verschoben werden. Wenn z.B. ein Online-Druckdienst die Fotos seiner Kunden auf Grund schnell anwachsenden Speicherplatzbedarfs nicht mehr lokal speichern kann, können die bestehenden Fotos in einen Cloud BLOB Datenspeicher verschoben werden und neu hochgeladene Fotos zuerst in diesem BLOB Datenspeicher gespeichert werden. Erst wenn die Fotos für den Druck benötigt werden, können sie heruntergeladen werden und nach der Verarbeitung vom lokalen Speicher wieder gelöscht werden. Ebenso kann ein Cloud NoSQL Datenspeicher die Meta-Daten zu den Fotos oder Bestellaufträge speichern.

Datennutzung aus der Cloud

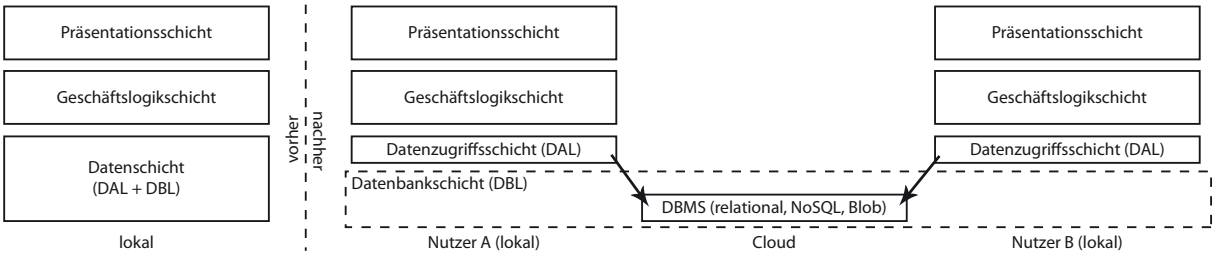

Abbildung 17 Vorher-Nachher-Vergleich einer Datennutzung aus der Cloud

Eine Untergruppe des reinen Outsourcings des DBL ist die Datennutzung aus der Cloud. Hier wird, wie in Abbildung 17 zu sehen ist, der DBL einer Anwendung in die Cloud migriert und die lokale Anwendung greift zukünftig nicht mehr auf den lokalen DBL zu, sondern auf den entfernten DBL in der Cloud. Die damit einhergehende höhere Latenz spielt für bestimmte Anwendungen keine große Rolle.

Ein praktisches Szenario für die alleinige Migration des DBL in die Cloud sind Anwendungen die nicht nur für eine Abteilungen relevant sind, aber aktuell innerhalb einer Abteilung betrieben werden (Schatten-IT [125]) oder auf kleinen Servern im lokalen Rechenzentrum betrieben werden. Hier kann der DBL in die Cloud migriert werden und die lokale Anwendung kann mit geringer Anpassung weiter benutzt werden. Somit ist die Datenbank zusätzlich für andere Abteilungen eines Unternehmens leichter zugänglich und die Kosten für die Administration der Datenbanken können reduziert werden oder die Sicherheit auf Grund von häufig ausbleibenden Updates in einer Schatten-IT kann erhöht werden. [42]

4.1.3 Verwendung von Hochskalierbaren Datenspeichern (NoSQL, BLOB Datenspeicher)

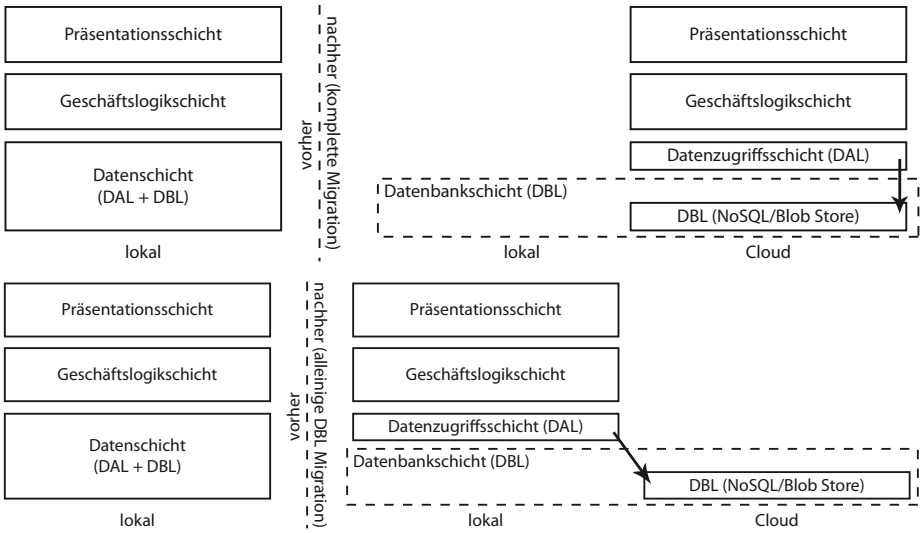

Abbildung 18 Vorher-Nachher-Vergleich einer Verwendung von hochskalierbaren Datenspeichern

Bei der Verwendung von hochskalierbaren Datenspeichern, wie in Abbildung 18 gezeigt, wird davon ausgegangen, dass zuvor ein RDBMS benutzt wurde, welches aber nicht hochskalierbar ist und die Daten nun in einen hochskalierbaren Cloud Datenspeicher migriert werden sollen. Dabei kann das genutzte RDBMS vor der Migration sowohl in der Cloud, als auch lokal laufen.

Diese Möglichkeit der Datenmigration ist, wie in Abschnitt 2.4 erläutert, nicht immer möglich, da hochskalierbare Datenspeicher nicht die gleiche Funktionalität wie RDBMS besitzen. So ist es nicht möglich im gleichen Umfang Transaktionen, die die ACID Kriterien erfüllen, gegen z.B. NoSQL Datenbanken laufen zu lassen und komplexe SQL-Abfragen mit Joins sind auch häufig nicht möglich und müssen in der Anwendungslogik umgesetzt werden [126]. Die zu migrierende Anwendung darf also nicht auf (komplexen) Transaktionen und komplizierten Abfragen beruhen. Außerdem muss sowohl die Datenschicht, als auch die darüber liegende Anwendungsschicht in den meisten Fällen erheblich angepasst werden um die Daten zukünftig in hochskalierbaren Datenspeichern zuverlässig abzulegen.

Neben der kompletten Datenmigration in einen hochskalierbaren Datenspeicher besteht auch die Möglichkeit nur bestimmte Daten z.B. in einer NoSQL Datenbank abzulegen [35]. Damit können diejenigen Daten, die Bestandteil von Transaktionen sind, weiter in einem RDBMS gehalten werden (Polyglot Persistence [7]).

Zwei passende Beispiele aus der Praxis sind Netflix [127] und Alexa [128]. Beide Firmen haben Daten aus einem RDBMS in einen Cloud NoSQL und BLOB Datenspeicher (Amazon SimpleDB und S3) migriert um eine bessere Skalierbarkeit zu erreichen.

4.1.4 Geographische Replikation

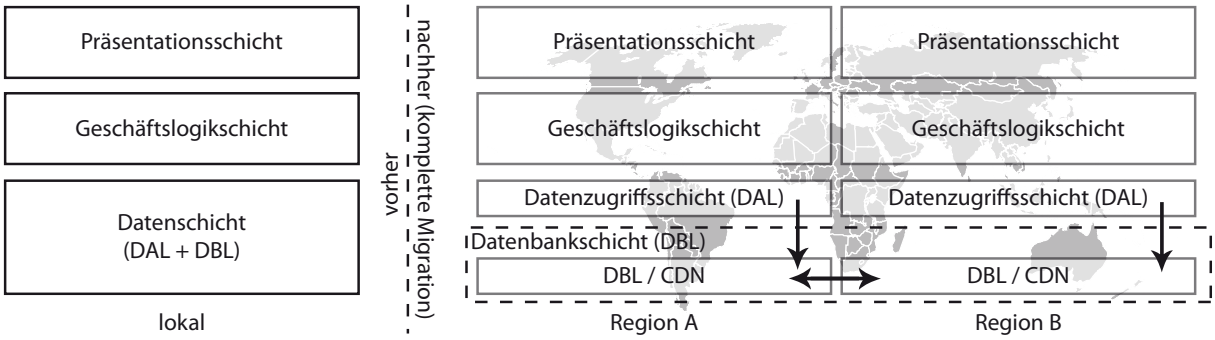

Abbildung 19 Vorher-Nachher-Vergleich einer geographischen Replikation, Quelle der Weltkarte [129]

Um die Latenz beim Zugriff auf Daten möglichst gering zu halten wird oft versucht die Daten zum einen möglichst nah an die Berechnung zu bringen und zum anderen möglichst nah an den Benutzer. Dies ist mit der Replikation der Datenschicht und der dazugehörigen höheren Anwendungsschichten in für die Benutzer naheliegende Cloud Rechenzentren, wie in Abbildung 19 gezeigt, möglich. Zusätzlich bedeutet dieser Ansatz, dass die Daten zwischen den verschiedenen Replikas synchron gehalten werden müssen. [90]

Geographische Replikation kann sowohl für statische Daten, in Form eines CDN, als auch für dynamische Daten in Form von Lese-Repliken (Master/Slave Replikation) oder Lese/Schreib-Repliken (Master/Master Replikation) geschehen [126].

Ein praktisches Beispiel für ein CDN ist Amazon CloudFront [130] und für Datenbankreplikation Azure SQL mit der Data Sync Funktion [43]. Da ein CDN prinzipiell ein Verteilungsdienst rund um einen BLOB Datenspeicher ist, wird das CDN nicht separat als Quell- und Ziel-Datenspeichertyp aufgelistet.

Das Szenario Geographische Replikation fokussiert sich dabei auf das Kopieren und Synchronisieren der bestehenden Daten und beinhaltet keinen Wechsel des Datenspeichertyps. Dies kann z.B. durch die zusätzliche Auswahl des Szenarios Verwendung von hochskalierbaren Datenspeichern ermöglicht werden.

4.1.5 Datenverteilung (Sharding)

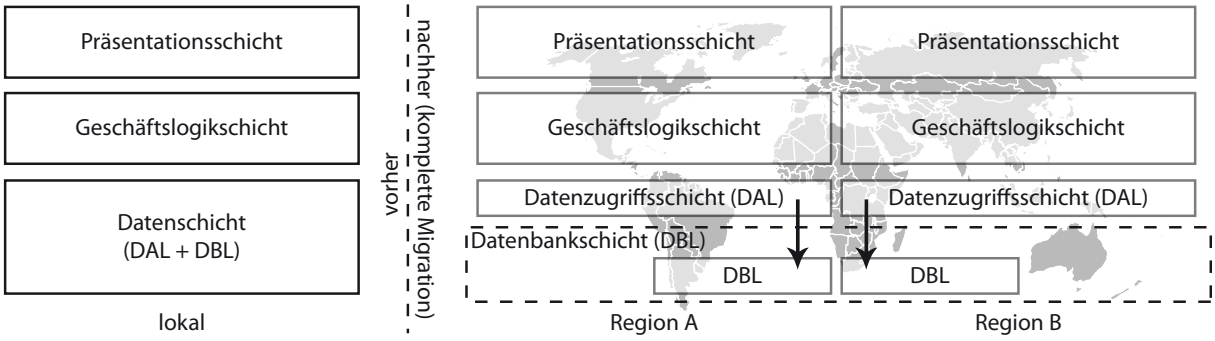

Abbildung 20 Vorher-Nachher-Vergleich einer Datenverteilung, Quelle der Weltkarte [129]

Ähnlich zur geographischen Replikation können Daten anstatt kopiert zu werden auch auf verschiedene Rechenzentren, wie in Abbildung 20 gezeigt, verteilt werden. Auch dieser Ansatz kann Daten näher an Benutzer bringen und somit die Latenzen verringern. Außerdem ermöglicht dieser Ansatz eine bessere Skalierbarkeit, da die Daten nun über mehrere DB-Instanzen oder DB-Cluster verteilt werden können und somit Limits wie z.B. max. 1 TB Speicher bei Amazon RDS umgangen werden können.

Die Verteilung der Daten kann entweder geographisch geschehen um die Daten näher an Benutzer zu bringen, oder, wenn dies nicht sinnvoll erscheint auch nach Funktion der Daten (Sharding) [85]. Die verteilten Daten sind dabei disjunkt und es erfolgt keine Synchronisation der Shards.

Praktische Beispiele inklusive Anleitungen für die Implementierung gibt es z.B. für SQL Azure [131] und Amazon RDS [132].

4.1.6 Off-Loading of Peak Loads (Cloud Burst)

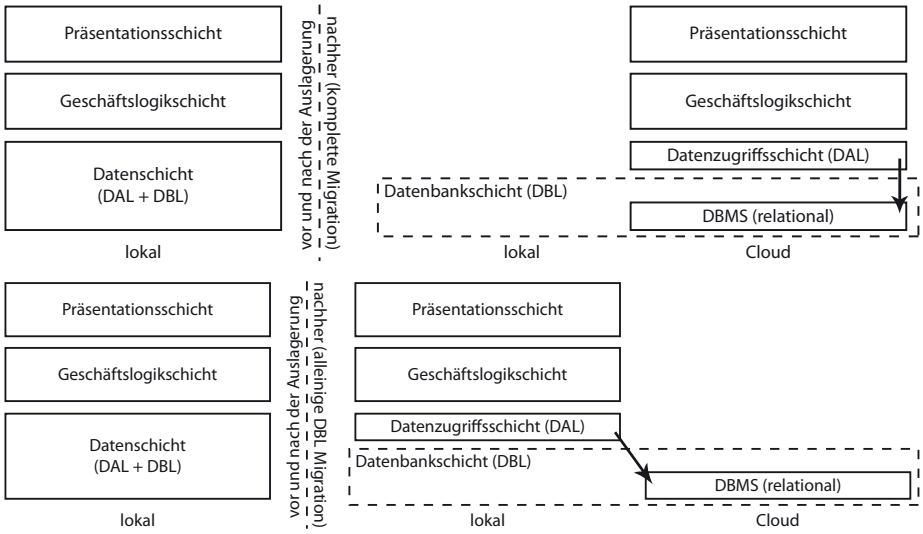

Abbildung 21 Vorher-Nachher-Vergleich eines Cloud Burst Szenarios

Off-Loading of Peak Loads, auch Cloud Burst [10] genannt, ist die temporäre Auslagerung von Infrastruktur in die Cloud, wie in Abbildung 21 und [43] gezeigt. Ursache dafür ist i.d.R. ein zeitlich begrenzter sehr hoher Ressourcenbedarf, der nicht mit den Ressourcen des lokalen Rechenzentrums bewerkstelligt werden kann. Dazu werden entweder Ressourcen aus der Cloud hinzugeschaltet und übernehmen somit die zusätzliche Last oder die gesamte Anwendung wird temporär in die Cloud ausgelagert und läuft dort so lange, bis das lokale Rechenzentrum die Last wieder selbst bewerkstelligen kann.

Um die Latenz während der Auslagerung der Datenschicht weiterhin gering zu halten, muss in der Regel die Anwendung nah am DBL liegen und somit auch mit ausgelagert werden.

Häufig genannte Beispiele für hohe Last sind Saisongeschäfte wie Weihnachten oder Muttertag. Möglich ist Cloud Burst z.B. mit SQL Azures Data Sync Funktion [43].

4.1.7 Arbeiten auf Datenkopie (Datenanalyse und Monitoring)

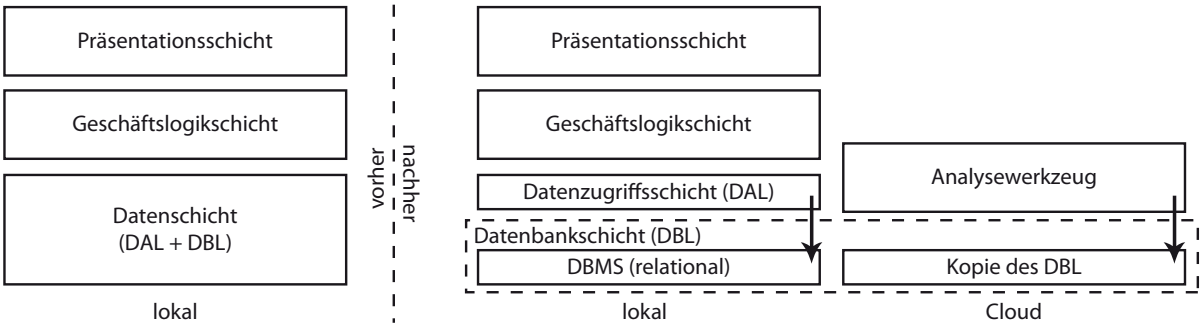

Abbildung 22 Vorher-Nachher-Vergleich der Kopie des DBL um auf dieser Datenkopie zu arbeiten

Beim Arbeiten auf einer Datenkopie, wie in Abbildung 22 gezeigt, wird eine reine Kopie des gesamten oder teilweisen DBL erstellt. Diese Kopie dient in der Regel der Entlastung von Produktivsystemen, wenn auf den gleichen Daten komplexe Analysen erstellt werden sollen. Hierbei ist es möglich eine Kopie einer lokalen Datenbank in die Cloud auszulagern und mit Analysewerkzeugen auf dieser Kopie zu arbeiten. Der Typ des Quell- und Zieldatenspeichers ist i.d.R. ein RDBMS, da NoSQL und BLOB Datenspeicher häufig selbst sehr gut skalieren können und somit auch rechenintensive Analysen parallel zum Produktionsbetrieb verkraften.

Sofern die Daten schützenswert sind und sich die Kopie des DBL in einer Public Cloud befindet, oder die schützenswerten Anteile der Daten auf Grund z.B. von Privatheit oder Geschäftsgeheimnissen nicht von Dritten betrachtet werden sollen [18], bieten sich bestimmte Patterns für Vertraulichkeit an. Strauch et al. identifizieren dazu in [4] die Patterns Confidentiality Level Data Aggregator, Confidentiality Level Data Splitter, Filter of Critical Data, Pseudonymizer of Critical Data und Anonymizer of Critical Data.

Ein praktisches Beispiel sind Datawarehose Anwendungen, die nicht auf den Live-Daten arbeiten.

4.1.8 Datensynchronisation

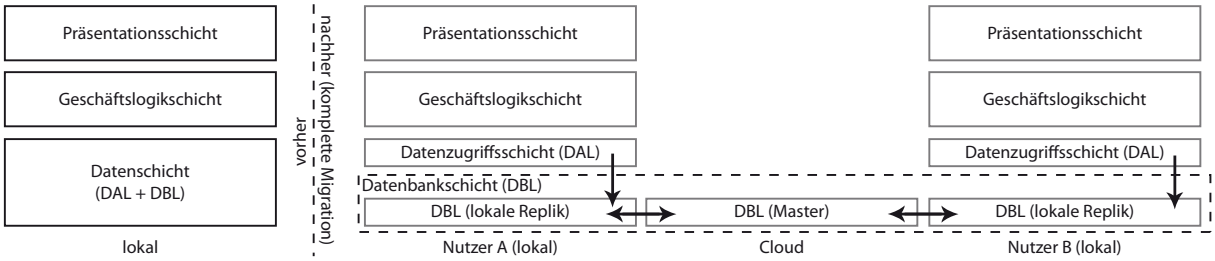

Abbildung 23 Vorher-Nachher-Vergleich eines Datensynchronisationsszenarios

Die Datensynchronisation mit der Cloud, wie in Abbildung 23 gezeigt, ermöglicht es mehreren Nutzern auf relativ aktuellen Daten offline zu arbeiten. Dazu wird der bisher lokale DBL in die Cloud kopiert und danach eine Synchronisation eingerichtet. Damit ermöglicht man die Daten mehreren Nutzern parallel zur Verfügung zu stellen, zeitweise offline zu gehen und trotzdem ein relativ aktuelles Bild des Gesamtdatenbestandes zu haben. Es wird auf einer lokalen Replik der Daten gearbeitet, die regelmäßig mit der Cloud Datenbank synchronisiert wird. Damit erhöht sich die Latenz nicht beim Zugriff auf die Daten und dennoch stehen die Änderungen nach kurzer Zeit allen verbundenen Nutzern zur Verfügung.

Ein praktisches Beispiel dazu wurde bereits in Abschnitt 2.1.2 erläutert.

4.1.9 Backup

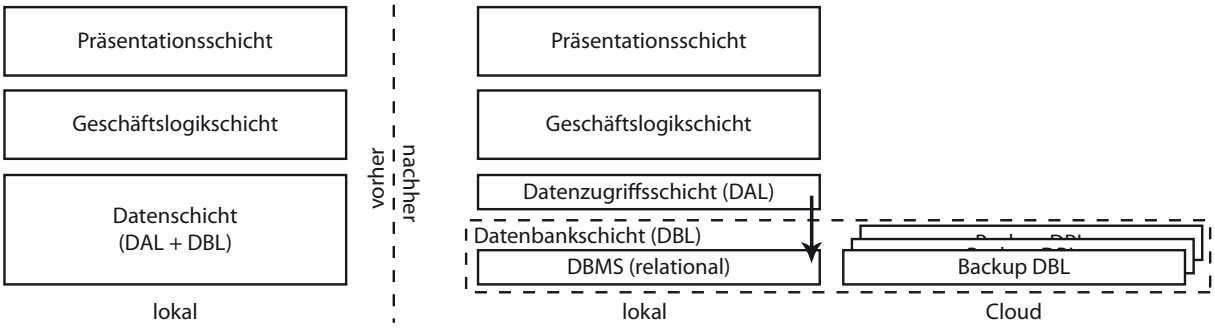

Abbildung 24 Vorher-Nachher-Vergleich eines entfernten Backups

Ein Backup, wie in Abbildung 24 gezeigt, beinhaltet die reine Kopie des DBL z.B. in die Cloud und ist damit unabhängig von der gesamten Migration einer Anwendung in die Cloud. Das Backup hält dann einen Zustand zu einem bestimmten Zeitpunkt fest und dient der Sicherheit um bei Ausfällen die Daten wiederherzustellen. Verschiedene Gesetze und Richtlinien wie Gramm-Leach-Bliley Act of 1999 [133] oder SOX [134] schreiben Backups von Daten für eine bestimmte Zeit vor. Besonders bei der Nutzung von Cloud Diensten wie SaaS wird verlangt, regemäßig Backups der online gespeicherten Daten anzufertigen und lokal zu speichern, da man selbst keine Kontrolle über die Daten innerhalb der Cloud Dienste hat. [66]

Auch für externe Backup-Lösungen kann die Cloud als günstige Variante genutzt werden, sofern die Regularien dies zulassen [62].

Ein praktisches Beispiel ist Datacastle [135], die eine Cloud Backup Lösung auf Microsoft Azure Basis anbieten, oder der Data Sync Dienst von SQL Azure selbst [43].

4.1.10 Archivierung

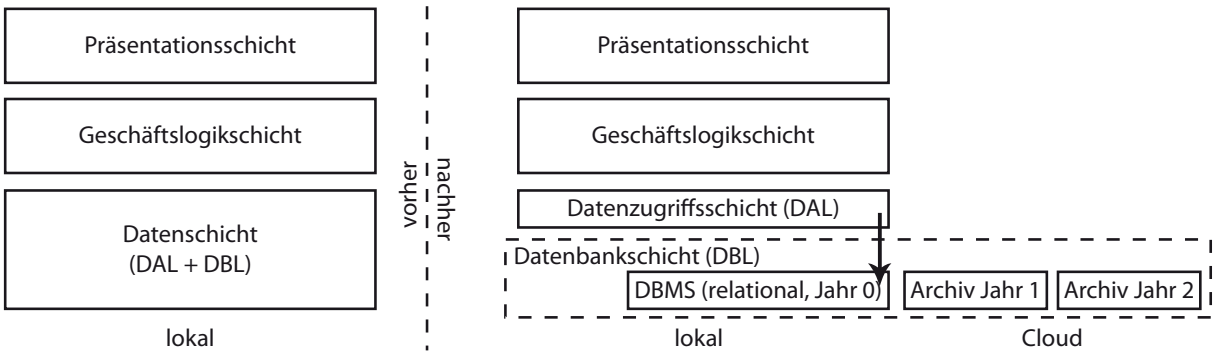

Abbildung 25 Vorher-Nachher-Vergleich eines entfernten Archivs

Archivierung ist, wie in Abbildung 25 gezeigt, ähnlich dem Backup auch eine reine Kopie des gesamten oder teilweisen DBL, erfüllt aber einen anderen Zweck. Zum einen müssen aus regulatorischen Gründen bestimmte Daten für eine bestimmte Zeit aufbewahrt werden und zum anderen können Daten, die im Produktivsystem nicht mehr benötigt werden, entfernt werden um mehr Platz für neue Daten zu schaffen. Eine Wiederherstellung von archivierten Daten ist nicht für den Fehlerfall gedacht, sondern zum einen für Analysen und zum anderen für Beweise z.B. in gerichtlichen Prozessen. [136]

Ein praktisches Beispiel ist AWS welche mit S3 einen zertifizierten Dienst auch für Archivierungen anbieten. Auf S3 aufbauend haben sich weitere Anbieter speziell auf Archivierungslösungen spezialisiert. Neben BLOB Datenspeichern wie Amazon S3 gibt es auch spezielle Lösungen um Daten günstig zu archivieren: Dazu muss allerdings in Kauf genommen werden, dass die archivierten Daten explizit angefordert werden müssen und erst nach einer gewissen Wartezeit zur Verfügung stehen (Amazon Glacier [22]). [137]

4.1.11 Datenimport aus der Cloud

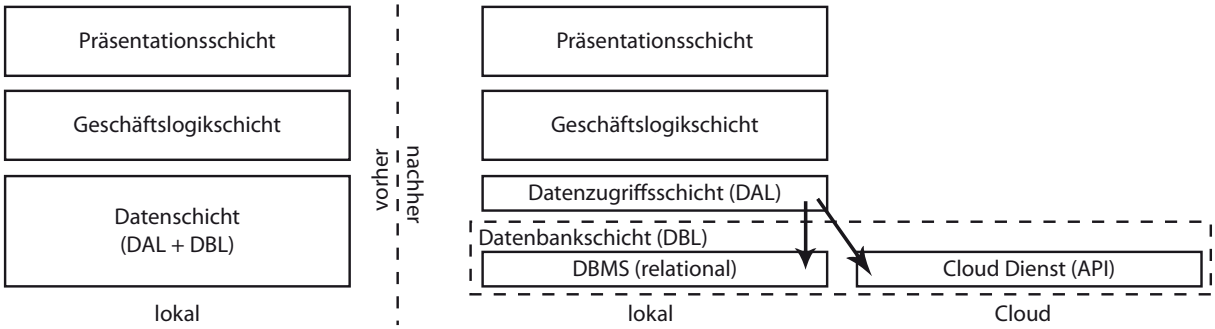

Abbildung 26 Vorher-Nachher-Vergleich eines Datenimports aus der Cloud

Einige Cloud Dienste stellen ihren Daten über eine API zum Abruf zur Verfügung. Abbildung 26 zeigt, wie über diese Programmierschnittstellen Daten aus Cloud Diensten abgerufen und lokal verarbeitet, sowie gespeichert werden können. Hierbei wird also der gesamte oder Teile des DBL eines Cloud Dienstes in das lokale Rechenzentrum importiert um mit geringer Latenz auf den Daten zu arbeiten.

Beispiele für solche Anbieter sind Regierungen, die ihre Daten maschinenzugänglich veröffentlichen (Open Data [138]) und Cloud Dienste die Daten zur Verfügung stellen, wie z.B. Twitter Streams [139] um Social Media Monitoring [140] zu betreiben.

4.2 Taxonomie der Migrationsszenarien

Im Folgenden werden allgemeine Kriterien beschrieben (Abschnitt 4.2.1), die bei der Cloud Datenmigration eine Rolle spielen. Im Anschluss werden diese Kriterien auf die oben identifizierten Cloud Datenmigrationsszenarien abgebildet und die Abbildungen begründet (Abschnitte 4.2.2 ff).

4.2.1 Überblick Kriterien der Migrationsszenarien (Datenmigrationsstrategie)

Die hier beschriebenen Kriterien zur Cloud Datenmigration wurden aus Kapitel 2 und Abschnitt 4.1 extrahiert und werden kurz erläutert. Die *kursiv* geschriebenen Wörter heben unterschiedliche Ausprägungen einer Kategorie hervor, welche in der Abbildung auf die Cloud Datenmigrationsszenarien eine Rolle spielen.

Ablösung

Prinzipiell kann eine Datenmigration auf zwei Arten geschehen, mit und ohne Unterbrechung des Betriebs. Wenn der Betrieb nicht unterbrochen werden soll, spricht man von einer *Live-Migration* oder Online-Migration, wenn eine Unterbrechung vorgesehen ist von einer *Non-Live-Migration* oder Offline Migration. [17], [141]

Für große Datenmengen empfiehlt sich Changed Data Capture (CDC), welches zur Live-Migration gezählt werden kann. Hier wird ein Snapshot des Quellsystems in das Zielsystem migriert und die Änderungen nach dem Snapshot protokolliert. Dies geschieht ent-

weder per Transaktionslog ab dem Zeitpunkt des Snapshots oder unter Nutzung von Triggern und temporären Tabellen, die die Änderungen nach dem Snapshot festhalten. Die Größe der Änderungen ist wesentlich kleiner als der Snapshot und kann anschließend in einer sehr kurzen Downtime migriert werden. [17]

Verbleibende lokale Existenz des DBL
Abhängig vom Migrationsszenario kann die lokale Instanz des DBL weiter lokal verbleiben um als Masterdatenbank oder Replikat zu dienen (*Kopie* mit oder ohne *uni-/bidirektionale Synchronisation*). Oder aber der lokale DBL wird teilweise oder komplett in die Cloud *verschoben*. [42], (Abschnitte 4.1.2, 4.1.7, 4.1.8)

Grad der Migration
Die Daten aus dem Quellsystem können entweder *vollständig* oder nur *teilweise* in ein oder mehrere Zielsysteme migriert werden. Dabei können die Daten auch aus einem Quellsystem vollständig in unterschiedliche Zielsystemen aufgehen, die dann jeweils nur einen Teil der Daten des Quellsystems enthalten. Eine vollständige Migration bedeutet nicht, dass auch alle Daten auf einmal migriert werden (siehe Zeitspanne der Migration), sondern beinhaltet lediglich die Absicht den gesamten DBL zu migrieren. [35]

Quell- und Ziel-Datenspeichertyp
Als Datenspeichertypen werden in dieser Arbeit RDBMS, NoSQL und BLOB Datenspeicher unterschieden. Bei der Datenmigration spielt es eine Rolle ob gleichzeitig ein Wechsel des Datenspeichertyps vollzogen wird. So können z.B. Daten aus einem *RDBMS* zu einer Kombination aus *NoSQL* Datenspeicher, *BLOB* Datenspeicher und RDBMS migriert werden, was einen komplexen ETL-Prozess beinhaltet aber im Nachhinein auch wesentliche Skalierbarkeitsvorteile mit sich bringen kann. CDNs für statische Daten werden nicht separat als Datenspeichertyp aufgefasst, da sie die Daten eines BLOB Datenspeichers auf andere Knoten eines BLOB Datenspeichers verteilen und sich ansonsten aber vom Datenspeichertyp her nicht von BLOB Datenspeichern unterscheiden. [35] [130]

Versions-/Produktwechsel des DBL
Wenn der Quell- und Ziel-Datenspeichertyp (z.B. RDBMS) der gleiche ist, kann es dennoch Unterschiede in der *Version* des verwendeten Produkts geben oder sich um ein anderes *Produkt* des gleichen Datenspeichertyps handeln. Diese Versions- oder Produktwechsel erschweren die Migration, da bei einem Versionswechsel ggf. Migrationsskripte ausgeführt werden müssen oder bei Produktwechseln Inkompatibilitäten zwischen den Produkten bestehen. Mögliche Lösungen um solche Probleme zu kompensieren sind Adapter oder Emulatoren, welche Funktionen des vorherigen Produktes übernehmen, wie sie von Strauch et al. in [12] geschildert werden. (Abschnitt 2.2.1)

Richtung der Datenbewegung
Die klassische Cloud Datenmigration verschiebt *Daten in die Cloud*, es ist aber auch möglich *Daten aus der Cloud* in ein lokales Rechenzentrum zu verschieben oder *zwischen Clouds* hin- und her zu schieben. (Abschnitte 4.1.2, 4.1.8, 4.1.11)

In Abschnitt 4.2 wird davon ausgegangen, dass es keinen Unterschied macht, ob es sich bei einer Cloud um eine Private Cloud, Community, Cloud, Public Cloud oder Hybrid Cloud handelt. Eine differenziertere Betrachtung dazu wird in Abschnitt 4.3 vorgenommen.

Dauerhaftigkeit der Migration des DBL
Wie das Cloud Burst Szenario gezeigt hat, kann eine Migration auch geplant *temporär* erfolgen und muss nicht wie bei den anderen Cloud Datenmigrationsszenarien *dauerhaft* sein. (Abschnitte 4.1.6)

Zeitspanne der Migration
Bei der Planung der Migration muss vorab schon entschieden werden ob alle Daten auf einmal migriert werden sollen (*Big-Bang* Ansatz) oder ob die Migration *schrittweise* geschehen soll. Auch wenn eine vollständige Migration geplant ist (Grad der Migration), kann diese schrittweise erfolgen. (Abschnitt 2.2.1)

Variabilität der Ressourcennutzung des DBL
Um die Elastizitätseigenschaften der Cloud sinnvoll auszunutzen sollte die Nutzung des DBL auch variabel sein um die vollen Vorteile des Cloud Computing auszuschöpfen. Typische Eigenschaften, die eine Migration des DBL in die Cloud nahelegen, sind *zyklische* und *azyklische* Variabilität der Ressourcennutzung des DBL. Aber auch bei *statischem* Anstieg der Ressourcennutzung des DBL kann eine Migration in die Cloud sinnvoll sein, um z.B. die Daten nahe an einer Cloud-Anwendung zu halten oder aus eventuellen Kosteneinsparungen. (Abschnitt 4.1.6)

Anstieg der Ressourcenauslastung des DBL
Im Falle einer zyklischen oder azyklischen Variabilität der Ressourcennutzung des DBL kann zusätzlich noch unterschiedenen werden, ob der Anstieg an Ressourcenbedarf *schnell* oder *langsam* erfolgt. Bei einem langsamen Anstieg können in der Regel die zusätzlich benötigten Ressourcen bei Bedarf automatisch allokiert werden, bei einem erwarteten sehr schnellen Anstieg können vorsorglich genügend Ressourcen allokiert werden, bei unerwartetem sehr schnellen Anstieg des Ressourcenbedarfs müssen allerdings ständig zusätzliche Ressourcen allokiert sein um nicht die Antwortzeiten extrem zu erhöhen oder die Systemstabilität zu gefährden. Die Auswahl des Cloud Anbieters spielt hier eine große Rolle, da die verschiedenen Anbieter unterschiedlich schnell neue Ressourcen allokieren können.

Kriterium/ Szenario	**Reines Outsourcing des DBL**	**Hochskalierbare Datenspeicher**	**Geographische Replikation**	**Sharding**	**Cloud Burst**	**Arbeiten auf Datenkopie**	**Datensynchronisation**	**Backup**	**Archivierung**	**Datenimport aus der Cloud**	**Datennutzung aus der Cloud**
Ablösung											
Live (ohne Downtime)	x	x	x	x	x	x	x	x	x	x	x
Non-Live (mit Downtime)	x	x	x	x	x	x	x	-	-	x	x
Verbleibende lokale Existenz des DBL											
Verschieben des DBL in die Cloud	x	x	x	x	x	-	-	-	x	-	x
Kopie des DBL in die Cloud	-	-	x	-	x	x	x	x	x	-	-
Einweg-Synchronisation	-	-	x	-	x	-	x	-	-	x	-
Zweiweg-Synchronisation	-	-	x	-	x	-	x	-	-	-	-
Grad der Migration											
vollständig	x	(x)	x	x	x	x	x	x	x	x	x
teilweise	x	x	x	x	x	x	x	x	x	x	x
Quell-Datenspeichertyp											
RDBMS	x	x	x	x	x	x	x	x	x	x	x
NoSQL	x	-	x	x	x	(x)	x	x	x	x	x
BLOB	x	-	x	x	x	(x)	x	x	x	x	x
Ziel-Datenspeichertyp											
RDBMS	x[2]	x[1]	x[2]	x	x[2]	x	x	x[2]	x	x	x[2]
NoSQL	x[2]	x	x[2]	x	x[2]	(x)	x	x[2]	x	x	x[2]
BLOB	x[2]	x	x[2]	x	x[2]	(x)	x	x	x	x	x[2]
Versions-/Produktwechsel des DBL											
gleiches Produkt, unterschiedliche Version	x	-	x	x	x	x	x	x	x	-	x
gleiches Produkt, gleiche Version	x	-	x	x	x	x	x	x	x	-	x
unterschiedliches Produkt	x	x	x	x	x	x	x	x	x	-	x
Richtung der Datenbewegung											
von lokalem Rechenzentrum in die Cloud	x	x	x	x	x	x	x	x	x	-	x
von Cloud in lokales Rechenzentrum	x	x	x	x	x	x	x	x	x	x	-
zwischen Clouds	x	x	x	x	(x)	x	x	x	x	x	x
Dauerhaftigkeit der Migration des DBL											

[1] teilweise ist es sinnvoll nicht alle Daten in einen NoSQL oder Blob Store zu migrieren und Daten, die innerhalb einer Transaktion verwendet werden, im RDBMS zu belassen

[2] sofern der gleiche Datenspeichertyp als Quelle ausgewählt wurde, da in diesem Szenario kein Wechsel des Datenspeichertyps vorgesehen ist

dauerhaft	x	x	x	x	-	x	x	x	x	x	x
temporär	-	-	-	-	x	-	-	-	-	-	-
Zeitspanne der Migration											
Stichtagsumstellung (Big-Bang)	x	x	x	x	x	x	x	x	x	x	x
längerer Zeitraum (schrittweise)	x	x	x	x	-	x	x	x	x	x	x
Variabilität der Ressourcennutzung des DBL											
zyklisch variabel	x	x	x	x	x	-	x	-	-	x	x
azyklisch variabel	x	x	x	x	x	-	x	-	-	x	x
statisch	x	x	x	x	-	-	x	-	-	x	x
Anstieg der Ressourcenauslastung des DBL											
schnell	x	x	x	x	x	-	x	-	-	x	x
langsam	x	x	x	x	x	-	x	-	-	x	x

Tabelle 1 Abbildung der Kriterien zur Cloud Datenmigration auf Cloud Datenmigrationsszenarien (Migrationsstrategie)

Tabelle 1 beschreibt welche Ausprägungen die jeweiligen Cloud Datenmigrationsszenarien in den jeweiligen Kriterien erreichen können. Eine Unterstützung einer Ausprägung eines Kriteriums im jeweiligen Szenario ist mit „x“ gekennzeichnet, für eine sinnvolle und vorstellbare, aber aktuell untypische Ausprägung steht „(x)“ und für unmögliche oder sehr untypische Ausprägungen im jeweiligen Szenario steht ein „-“.

In den folgenden Abschnitten werden die Abbildungen kurz begründet. Als Grundlage wird hier das reine Outsourcing des DBL genutzt und für die weiteren Cloud Datenmigrationsszenarien nur noch die Abweichungen zum reinen Outsourcing des DBL bezüglich der Abbildung der Kriterien beschrieben.

4.2.2 Reines Outsourcing des DBL

Ablösung

Als Ablösungsmöglichkeiten kommen beim reinen Outsourcing des DBL sowohl die Live-, als auch die Non-Live Migration in Frage, da bei kritischen Anwendungen z.B. die Downtime so gering wie möglich sein muss um keine wirtschaftlichen Konsequenzen durch Nichterreichbarkeit zu erleiden. Bei unkritischen Anwendungen kann eine einfachere Non-Live Migration z.B. in einem vorgesehen Wartungsfenster oder bei geringer/keiner Last (ggf. nachts) durchgeführt werden.

Amazon RDS bietet zum Beispiel die Möglichkeit im ersten Schritt einen aktuellen Snapshot seiner lokalen MySQL Datenbank nach RDS zu importieren (große Datei, zeitaufwändig) und im zweiten Schritt für die Transaktionen, die nach den Snapshot geschehen, die Log-Dateien zu importieren (wenige Daten, schnell möglich) und auf die importierten Daten anzuwenden. Ab Beginn des zweiten Schritts dürfen keine weiteren Schreibzugriffe auf die lokale MySQL Datenbank mehr zugelassen werden. Sobald auch die Transkations-Logs importiert wurden, kann der DNS-Eintrag (CNAME) für den MySQL Datenbankserver auf den RDS Server umgeleitet werden und sobald dieser durch das Netzwerk propagiert wurde, werden alle neuen Anfragen vom RDS Server entgegengenommen. Die dadurch entstandene Downtime ist minimal und kann je nach Größe der Transaktions-Logs und der Time-to-Live (TTL) Einstellung des DNS Caches nur wenige Minuten betragen. [142]

Verbleibende lokale Existenz des DBL
Offensichtlich handelt es sich beim reinen Outsourcing des DBL um eine Verschiebung des DBL, ohne dass langfristig auf einer Kopie des DBL gearbeitet oder eine Synchronisation benötigt wird.

Grad der Migration
Besonders bei personenbezogenen Daten oder Geschäftsgeheimnissen muss geprüft werden, ob auch diese Daten ohne Einschränkung in die Cloud migriert werden können (siehe Abschnitt 1.3). Es ist also beim reinen Outsourcing des DBL möglich sowohl den gesamten DBL zu migrieren oder nur einzelne Tabellen eines RDBMS. Bei einer teilweisen Migration ist allerdings die Auswirkung auf den DAL wesentlich höher als bei einer vollständigen Migration, da hier der DAL wissen muss, welche Daten wo gespeichert sind und wie auf diese zugegriffen werden können.

Quell- und Ziel-Datenspeichertyp
Häufig handelt es sich bei der Migration von bestehenden Anwendungen um RDBMS und BLOBs die in die Cloud migriert werden sollen, da die Verbreitung von NoSQL Datenbanken noch relativ gering ist [7]. Prinzipiell können aber beim reinen Outsourcing des DBL jegliche Quell-Datenspeichertypen migriert werden, solange es sich auf der Zielplattform um den gleichen Datenspeichertyp handelt.

Sollte ein Wechsel z.B. von einem RDBMS zu einem NoSQL Datenspeicher vollzogen werden, wird dies im Rahmen dieser Arbeit nicht als reines Outsourcing des DBL betrachtet, sondern um eine Migration in einen hochskalierbaren Datenspeicher.

Versions-/Produktwechsel des DBL
Solange es sich bei einer Datenmigration um den gleichen Quell- und Ziel-Datenspeichertyp handelt, sind zumindest bei RDBMS oft Migrationen auch über verschiedene Produkte hinweg möglich. Allerdings ist dann eine Live-Migration wesentlich aufwändiger, da kein reiner Snapshot und Transaktions-Log in das Zielsystem importiert werden kann, sondern ein Migrationsassistent die Daten ggf. erst konvertieren muss.

Richtung der Datenbewegung
Da diese Arbeit sowohl die Migration des DBL in die Cloud, als auch aus der Cloud und zwischen Clouds betrachtet, können beim reinen Outsourcing des DBL jegliche Richtungen genutzt werden.

Dauerhaftigkeit der Migration des DBL
Das reine Outsourcing des DBL ist im Rahmen dieser Arbeit mindestens eine mittelfristige Maßnahme und damit als dauerhaft anzusehen. Im Kontrast dazu steht Cloud Burst, welches nur eine temporäre Migration des DBL beinhaltet.

Zeitspanne der Migration
Ähnlich zum Grad der Migration kann auch bei der Zeitspanne der Migration der gesamte DBL auf einmal migriert werden, oder aber Teile des DBL werden schrittweise migriert. Handelt es sich um eine schrittweise Migration, muss parallel wie bei der teilweisen Migration der DAL angepasst werden.

Variabilität der Ressourcennutzung des DBL und Anstieg der Ressourcenauslastung des DBL
Der Grund für ein reines Outsourcing des DBL kann sein, dass momentan im lokalen Rechenzentrum die Kapazität für die maximale Last ausgelegt ist und diese aber nur sehr selten erreicht wird. Die Migration des DBL in die Cloud kann damit existierende Kapazitäten besser ausnutzen und generell durch ihre Elastizitätseigenschaft und hohen Automatisierungsstand kostengünstiger sein.

Untergruppe Datennutzung aus der Cloud
Bei der Untergruppe des reinen Outsourcings „Datennutzung aus der Cloud" ist die Richtung der Datenbewegungen während der Migration entweder aus dem lokalen Rechenzentrum in die Cloud oder von einer Cloud zu einer anderen Cloud, aber niemals von einer Cloud in ein lokales Rechenzentrum. Ansonsten sind die gleichen Strategien wie beim reinen Outsourcing möglich.

4.2.3 Verwendung von Hochskalierbaren Datenspeichern (NoSQL, BLOB Datenspeicher)

Ablösung
Die Migration von einem RDBMS in einen hochskalierbaren Datenspeicher kann ebenfalls Live und Non-Live geschehen, allerdings gestaltet sich die Live-Migration wesentlich schwieriger, da ein ETL Prozess benötigt wird um die Daten z.B. in einer NoSQL Datenbank abzulegen.

Grad der Migration
Besonders bei der Migration von einem RDBMS in einen hochskalierbaren Datenspeicher wird es in der Regel nicht möglich oder notwendig sein alle Daten z.B. in eine NoSQL Datenbank zu migrieren, siehe Abschnitt 2.4.

Quell- und Ziel-Datenspeichertyp
Ein wichtiges Unterscheidungsmerkmal des Szenarios Verwendung von hochskalierbaren Datenspeichern ist, dass als Quell-Datenspeichertyp ein RDBMS existiert und als Ziel-Datenspeichertyp keine RDBMS in Frage kommen, zumindest nicht für den Teil der Daten der hochskalierbar sein soll, siehe Abschnitt 2.4.

Es ist auch möglich einen lokal existierenden hochskalierbaren Datenspeicher in die Cloud zu migrieren, hierbei handelt es sich dann allerdings um ein reines Outsourcing des DBL.

Versions-/Produktwechsel des DBL
Bei der Verwendung von hochskalierbaren Datenspeichern ist zwingend ein Produktwechsel erforderlich, da beim Wechseln von einem RDBMS zu einem hochskalierbaren Datenspeicher auch ein Wechsel des Datenspeichertyps vorliegt.

Variabilität der Ressourcennutzung des DBL und Anstieg der Ressourcenauslastung des DBL
Zusätzlich zu den Argumenten beim reinen Outsourcing des DBL ist ein Argument für die Verwendung von hochskalierbaren Datenspeichern die wesentlich bessere Skalierbarkeit.

4.2.4 Geographische Replikation

Ablösung

Wenn eine geographische Replikation live geschehen soll, müssen zuerst die Repliken in den anderen geographischen Regionen erstellt und eine Synchronisation untereinander eingerichtet werden. Hat das gesamte System einen stabilen Zustand erreicht, d.h. vollständig synchronisiert, können DNS Server so konfiguriert werden, dass sie z.B. basierend auf Latenzzeiten die IP Adresse des Servers mit der geringsten Latenz vom aktuellen Standort zurückgeben, was in der Regel die IP Adresse des am nächsten gelegenen Servers ist.

Verbleibende lokale Existenz des DBL

Der lokale DBL kann sowohl als Endpunkt im Netzwerk der Replikate agieren, als auch selbst in die Cloud migriert und damit verschoben werden. Eine Synchronisation wird in jedem Falle benötigt, außer wenn die Daten voneinander unabhängig sind und jedem Client ein fester Endpunkt zugewiesen ist. Abhängig von der Replikation (Master/Slave und Master/Master) ist eine Einweg- oder Zweiweg-Synchronisation notwendig [126].

4.2.5 Datenverteilung (Sharding)

Ablösung

Bei der Aufsplittung des ursprünglichen DBL auf mehrere Server kann eine Live-Migration ähnlich zum reinen Outsourcing des DBL geschehen. Auch hier ist mit einer kurzen Downtime zu rechnen, da die Schreiboperationen für kurze Zeit blockiert werden müssen.

MySQL Cluster z.B. bietet Sharding an und ermöglicht auch im Betrieb neue Knoten hinzuzufügen und die Partitionierung anzupassen. Dazu müssen die Daten allerdings erst in eine MySQL Cluster Installation importiert werden. [143]

Verbleibende lokale Existenz des DBL

Der lokale DBL kann als ein Knoten des DBL bestehen bleiben, oder aber komplett ersetzt werden durch mehrere Knoten in der Cloud.

4.2.6 Off-Loading of Peak Loads (Cloud Burst)

Verbleibende lokale Existenz des DBL

Beim Cloud Burst kann der lokale DBL als Knoten weiter verwendet, oder aber der gesamte DBL in die Cloud migriert werden. In jedem Fall muss sichergestellt sein, dass vor der Zuschaltung der Cloud Ressourcen, alle Daten synchron sind. Wird der lokale DBL weiter verwendet, muss eine Zwei-Wege-Synchronisation mit dem Cloud DBL eingerichtet werden, ansonsten reicht sowohl für die Initiierung, als auch für die Terminierung der zugeschalteten Ressourcen eine temporäre Ein-Weg-Synchronisation.

Richtung der Datenbewegung

Beim Cloud Burst macht es in der Regel keinen Sinn die Daten von einer Public Cloud in eine andere Public Cloud zu migrieren, da eine Cloud an sich ja beliebig skalierbar und elastisch sein soll. Es sind allerdings Szenarien vorstellbar in denen hohe Lasten einer Private Cloud in einer Public Cloud abgefangen werden (Cloud Burst) um z.B. Kosten zu sparen oder nicht an Kapazitätsgrenzen einer Private Cloud zu gelangen.

Dauerhaftigkeit der Migration des DBL

Charakteristisch für Cloud Burst ist, dass die Migration der Daten in die Cloud nur temporär ist. Nur für den Zeitraum der Spitzenlasten werden zusätzliche Ressourcen aus

der Cloud benötigt, danach können diese wieder abgeschaltet werden. Wenn die Auslagerung von Dauer ist, handelt es sich um Reines Outsourcing des DBL.

Zeitspanne der Migration
Da Cloud Burst, wie in Abschnitt 4.2.1 definiert, eine regelmäßige und automatisierte Migration der Daten in die Cloud darstellt, handelt es sich hier stets um eine Stichtagsumstellung. Der Stichtag ist allerdings lastabhängig und damit dynamisch. Gleiches trifft auf die Migration zur Beendigung des Cloud Burst Falles zu.

Variabilität der Ressourcennutzung des DBL
Da Cloud Burst nur eingesetzt wird, wenn die Ressourcenauslastung stark schwankt, spielt sie im Falle einer statischen Ressourcenauslastung des DBL keine Rolle.

4.2.7 Arbeiten auf Datenkopie (Datenanalyse und Monitoring)

Verbleibende lokale Existenz des DBL
Beim Arbeiten auf einer Datenkopie wird ähnlich wie beim Backup eine Kopie des DBL erstellt, jedoch wird sie hier in der Regel in ein anderes System importiert, um mit diesen Daten direkt weiter arbeiten zu können, ohne das Produktivsystem zu belasten.

Variabilität der Ressourcennutzung des DBL und Anstieg der Ressourcenauslastung des DBL
Da es sich beim Arbeiten auf einer Datenkopie um eine Kopie des DBL handelt, spielt, wie beim Backup, die Variabilität der Ressourcennutzung und der Anstieg der Ressourcenauslastung keine Rolle.

4.2.8 Datensynchronisation

Verbleibende lokale Existenz des DBL
Anders als beim reinen Outsourcing des DBL bleibt bei der Datensynchronisation der lokale DBL bestehen und wird ähnlich wie bei der geographischen Replikation mit den entfernten Knoten synchronisiert. Allerdings können bei der Datensynchronisation die entfernten Knoten auch bewusst für einen gewissen Zeitraum nicht verfügbar sein und sich dann wieder mit dem Master synchronisieren.

4.2.9 Backup

Ablösung
Um einen konsistenten Zustand festzuhalten, muss ein Backup auf einem konsistenten Snapshot basieren. Daher handelt es sich beim Backup um eine Live-Migration. Außerdem soll die Erstellung eines Backups zu keiner merklichen Unterbrechung im Betrieb führen.

Verbleibende lokale Existenz des DBL
Charakteristisch für das Backup ist das Erstellen einer Kopie des lokalen DBL und des Speicherns dieser Kopie in der Cloud, der lokale DBL bleibt stets bestehen.

Quell- und Ziel-Datenspeichertyp
Beim Backup besteht sowohl die Möglichkeit die Daten aus dem Quellsystem in ein gleiches Zielsystem zu sichern, oder aber lediglich die Export- oder Dump-Funktion zu benutzen und das Backup als einfache Datei oder Menge an Dateien in der Cloud zu speichern. Bei der Verwendung eines DBMS in einer virtualisierten oder IaaS Cloud Umgebung kann auch ein Snapshot der gesamten virtuellen Maschine erstellt werden und in der Cloud gespeichert werden.

Richtung der Datenbewegung
Ein Backup kann sowohl für einen lokalen, als auch für einen Cloud DBL erstellt werden und sowohl lokal, als auch in der Cloud gespeichert werden. Um zum Beispiel die Zuverlässigkeit zu erhöhen, ist es oft sinnvoll Backups eines Cloud DBL in einer anderen Cloud oder lokal zu speichern, um im Falle eines Ausfalls des Cloud DBL, das Backup in einer anderen Cloud oder lokal einzuspielen und die Anfragen an den ausgefallenen DBL auf einen neuen DBL umzulenken.

Variabilität der Ressourcennutzung des DBL und Anstieg der Ressourcenauslastung des DBL
Für ein Backup spielt es keine Rolle, ob die Ressourcennutzung des zu sichernden DBL variabel ist oder schnell anwachsen kann. Generell sollte wenn möglich ein Backup aber dann erstellt werden, wenn wenig Last auf dem DBL ist, um die evtl. damit verbundene Downtime (um einen konsistenten Snapshot zu erhalten) minimal zu halten [142].

4.2.10 Archivierung
Die Archivierung gleicht dem Backup sehr, bis auf die im Folgenden beschriebenen Punkte.

Verbleibende lokale Existenz des DBL
Charakteristisch für ein Archiv ist, dass ein Teil der Daten aus dem Produktivsystem in ein anderes System verschoben wird. Daher handelt es sich beim Archiv um eine teilweise Verschiebung des DBL, z.B. in die Cloud.

4.2.11 Datenimport aus der Cloud
Verbleibende lokale Existenz des DBL
Beim Datenimport aus der Cloud spielt primär die Migration von Daten aus der Cloud in eine andere Cloud oder das lokale Rechenzentrum eine Rolle. Daher wird der lokale DBL auch nicht migriert oder kopiert, sondern Daten in den lokalen DBL importiert und ggf. auch Änderungen aus dem Cloud DBL nach einem initialen Import synchronisiert.

Versions-/Produktwechsel des DBL
Beim Datenimport aus der Cloud werden in der Regel proprietäre Schnittstellen (APIs) verwendet und bedingen eine komplexere Umwandlung in einen Zieldatenspeicher. Eine Ausnahme könnte das verwendete XML Format sein, in welchem manche APIs ihre Antworten ausliefern. Diese Antworten im XML Format könnten direkt z.B. in Datenbanken mit XML Unterstützung gespeichert werden. Dennoch wird davon ausgegangen, dass es sich nicht um vergleichbar Quell- und Zielspeicher handelt.

Richtung der Datenbewegung
Da es sich bei diesem Szenario um einen reinen Import von Daten aus der Cloud handelt, dient als Ziel lediglich eine weitere Cloud oder der lokale DBL.

4.3 Abbildung der Migrationsszenarien auf Cloud Data Hosting Solutions
Im Folgenden werden parallel zu Abschnitt 4.2 die Migrationsszenarien auf Cloud Data Hosting Solutions abgebildet (Abschnitt 4.2.1) und die Abbildung begründet (Abschnitte 4.2.2 ff).

4.3.1 Überblick Cloud Data Hosting Solutions

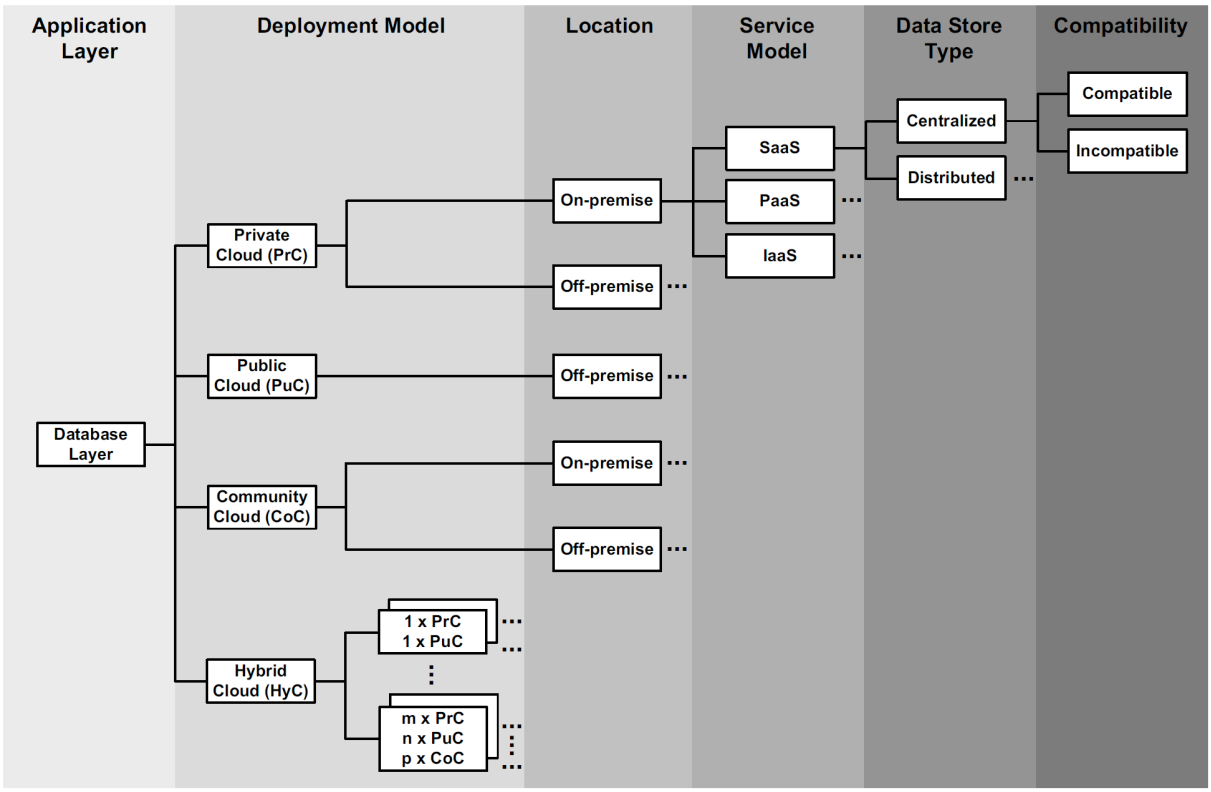

Abbildung 27 Taxonomie der Cloud Data Hosting Solutions [11]

Strauch et al. hat die in Abbildung 27 dargestellte Taxonomie für Cloud Data Hosting Solutions entwickelt, welche in den folgenden Abschnitten für die einzelnen Cloud Daten-Migrationsszenarien angewandt wird. Insbesondere werden hier die Auswirkungen der verschiedenen Cloud Deployment Modelle in Verbindung mit der Location und der Service Modelle auf die Szenarien untersucht. Sofern der Data Store Type (centralized oder distributed) des Ziel-Datenspeichers eine große Rolle hinsichtlich nichtfunktionaler Eigenschaften wie Verfügbarkeit oder Performanz spielt, wird auch dieser betrachtet. Das Kriterium Compatibility wurde schon durch das in dieser Arbeit verwendete Kriterium Versions-/Produktwechsel des DBL abgehandelt.

Cloud Data Hosting Solution / Szenario	**Reines Outsourcing des DBL**	**Hochskalierbare Datenspeicher**	**Geographische Replikation**	**Sharding**	**Cloud Burst**	**Arbeiten auf Datenkopie**	**Datensynchronisation**	**Backup**	**Archivierung**	**Datenimport aus der Cloud**	**Datennutzung aus der Cloud**
Deployment Model											
Private Cloud	x	x	(x)	x	x	x	x	x	x	x	x
Public Cloud	x	x	x	x	x	x	x	x	x	x	x
Community Cloud	x	x	x	x	x	x	x	x	x	x	x
Hybrid Cloud	x	x	x	x	x	x	x	x	x	x	x
Location											
On-Premise	x	x	(x)	x	x	x	x	x	x	x	x
Off-Premise	x	x	x	x	x	x	x	x	x	x	x
Service Model											
SaaS	-	-	-	-	-	-	-	x	x	x	-
PaaS	x	x	x	x	x	x	x	x	x	x	x
IaaS	x	x	x	x	x	x	x	x	x	x	x
Data Store Type											
Centralized	x	-	-	x	(x)	x	x	x	x	-	x
Distributed	x	x	x	x	x	x	x	x	x	-	x
Compatibility											
Compatible	x	x	x	x	x	x	x	x	x	-	x
Incompatible	x	x	x	x	x	x	x	x	x	x	x

Tabelle 2 Abbildung der Cloud Data Hosting Solution Kriterien auf Cloud Datenmigrationsszenarien

Tabelle 2 beschreibt welche Ausprägungen die jeweiligen Cloud Datenmigrationsszenarien in den jeweiligen Cloud Data Hosting Solution Kriterien erreichen können. Eine Unterstützung einer Ausprägung eines Kriteriums im jeweiligen Szenario ist mit „x“ gekennzeichnet und für eine unmögliche oder nicht sinnvolle Ausprägung im jeweiligen Szenario steht ein „-“. Ein „(x)“ kennzeichnet eine mögliche, aber untypische Ausprägung.

Der Autor dieser Arbeit stimmt nicht mit allen Zuordnungen zu den Service Models der in [11] genannten Beispiele überein. In dieser Arbeit werden die Zuordnungen bezüglich der Service Models aus Abschnitt 3.3 und 3.4 genutzt.

Um Wiederholungen zu vermeiden werden im folgenden Abschnitt die Abbildungen der Data Hosting Solutions auf das reine Outsourcing des DBL begründet und für die weiteren Cloud Datenmigrationsszenarien nur noch die Abweichungen zum reinen Outsourcing des DBL bezüglich der Abbildung der Cloud Data Hosting Solutions beschrieben.

4.3.2 Reines Outsourcing des DBL

Deployment Model und Location

Als Deployment Modell und damit auch der Location kommen beim reinen Outsourcing des DBL jegliche Formen in Frage. Für eine Entscheidung des Deployment Models sind hier u.a. die Kosten, Latenz und Vertraulichkeit der Daten relevant. Vertraulich Daten wie Geschäftsgeheimnisse oder personenbezogene Daten sollten nicht in einer Public Cloud gespeichert werden, wenn die Latenz eine große Rolle spielt, weil die höheren Anwendungsschicht nicht mit in die Cloud migriert werden können bietet sich eine Private, Community oder Hybrid Cloud an, die On-Premise betrieben wird. Public Clouds sind hingegen wegen ihrer besseren Skaleneffekte häufig kostengünstiger.

Insbesondere wenn unter anderen – aber nicht nur – Geschäftsgeheimnisse oder personenbezogene Daten betroffen sind, bietet sich das Hybrid Cloud Deployment Model an. So können die Geschäftsgeheimnisse oder personenbezogenen Daten in einer Private oder Community Cloud und die nicht sensiblen Daten z.B. in einer Public Cloud gehalten werden. Beispiele dafür sind Web-Shops oder öffentliche Verkehrsanbieter, die ihren Katalog bzw. Reiseverbindungs-Datenbank in einer Public Cloud, aber die Kunden und Bestelldaten in einer Private Cloud halten können. Da in der Regel wesentlich mehr Anfragen an einen solchen Katalog oder die Reiseverbindungs-Datenbank erfolgen, als an ein Kunden oder Buchungssystem, können somit mögliche Kosten- oder Performancevorteile durch geographische Replikation von Public Clouds besser genutzt werden.

Die Aufsplittung der Daten z.B. auf eine Public und Private Cloud bringt allerdings auch eine höhere Latenz mit sich und erfordert einen größeren Aufwand im Vergleich zur reinen Migration zu einem einzigen Deployment Model. So müssen der DAL und in der Regel auch die darüber liegenden Schichten erheblich angepasst werden. Zusätzlich muss überlegt werden, ob auch die höheren Schichten ganz oder teilweise in eine Public Cloud migriert werden können, da hier ggf. auch keine Geschäftsgeheimnisse oder personenbezogenen Daten verarbeitet werden dürfen. In einem solchen Fall spielen die in [4] vorgestellten Cloud Data Patterns zur Vertraulichkeit eine Rolle.

Weiterhin sind die Verantwortlichkeiten je nach Deployment und Service Model verschieden. Bei einer Private Cloud liegt die komplette Verantwortung bei einem selbst, bei der Community Cloud, bis hin zur Public Cloud nimmt die Verantwortung für die Verfügbarkeit der zu Grunde liegenden Infrastruktur und Dienste jeweils ab. Das gleiche gilt für die Service Modelle, hier hat man beim IaaS Modell die meiste Verantwortung. Das PaaS Modell bringt lediglich die Verantwortung für die eigene Software mit sich, welche beim SaaS Modell zusätzlich mit abgenommen wird. Die höchste Verantwortung, aber auch die meisten technischen Möglichkeiten, hat man also mit einer Private Cloud im IaaS Modell. Nutzt man hingegen einen SaaS Dienst aus einer Public Cloud, hat man kaum Einflussmöglichkeiten, aber auch keine Verantwortung bezüglich des Betriebs der Anwendung.

Service Model

Wie schon in Abschnitt 4.1.2 beschrieben, sind für das reine Outsourcing die Service Models IaaS und PaaS relevant. SaaS spielt hier in der Regel keine Rolle, da nicht genug Kontrolle oder Konfigurationsmöglichkeiten angeboten werden.

Beim PaaS Service Modell gibt es sowohl Dienste mit nativer (Amazon DynamoDB), als auch multiple instances (Amazon RDS) Mandantenfähigkeit. Diese Unterteilung ist hinsichtlich nichtfunktionaler Eigenschaften wie Daten- und Performance-Isolation oder Verfügbarkeit relevant. Bei nativer Mandantenfähigkeit nutzen verschiedene Nutzer die gleichen Anwendungsinstanzen, was in der Regel, je nach Last, zu unterschiedlichen Antwortzeiten und bei Sicherheitslücken leichter zur Sichtbarkeit der eigenen Daten für Dritte führt. Jedoch ist bei nativer Mandantenfähigkeit auch eine höhere Auslastung der Ressourcen und damit in der Regel niedrigere Kosten als bei der multiple instances Mandantenfähigkeit möglich. Mit multiple instances Mandantenfähigkeit hat jeder Nutzer zumindest seine eigene Anwendungsinstanz, teilweise auch seine eigene virtuelle Maschine (Amazon RDS) womit ggf. eine bessere Daten- und Performance-Isolation oder Verfügbarkeit erreicht werden kann. [2]

Das IaaS Service Modell bietet den Vorteil, dass die verwendete (virtualisierte) Hardware speziell auf das Anwendungsszenario abgestimmt werden kann. Für RDBMS spielt in der Regel die Arbeitsspeichergröße eine wichtige Rolle, wenn z.B. die gesamte Datenbank in den Arbeitsspeicher passt, erhöht sich der Performance enorm, da bei SQL-Anfragen nicht erst auf langsamere Festplatten zugegriffen werden muss. Populäre PaaS Dienste wie Amazon RDS ermöglichen zwar z.B. die Auswahl der Hardware, jedoch wird die maximale Speicher für alle Datenbanken auf insgesamt 1 TB beschränkt [99].

Data Store Type
Wenn die Möglichkeit zur Cluster-Konfiguration (von VM-Images) oder Redundanzeinstellungen (Multi-Region, Multi-Availability Zone, RAID-Verbund, etc.) im IaaS bzw. PaaS Service Modell möglich sind wird der Data Store Type als distributed bezeichnet, im anderen Fall als centralized.

Untergruppe Datennutzung aus der Cloud
Bezüglich der Cloud Data Hosting Solutions gibt es bei der Untergruppe Datennutzung aus der Cloud keine Unterschiede zum reinen Outsourcing des DBL.

4.3.3 Verwendung von Hochskalierbaren Datenspeichern (NoSQL, BLOB Datenspeicher)

Data Store Type
Charakteristisch für hochskalierbare Datenspeicher ist auch Hochverfügbarkeit, was durch Redundanz ermöglicht wurde und somit nur distributed Data Store Type zulassen.

4.3.4 Geographische Replikation

Deployment Model und Location
Als Deployment Model kommt bei der geographischen Replikation die On-Premise Private Cloud nur in dem Falle zum Einsatz, wenn sie als ein Knoten innerhalb des geographischen Clusters agiert. In der Regel ist es aber sinnvoller alle Knoten bei der geographischen Replikation in einer einheitlichen Cloud Umgebung zu betreiben um den Administrations- und Wartungsaufwand gering zu halten. Um die Ausfallsicherheit innerhalb einer Region [144] zu erhöhen können die Knoten innerhalb, der für die geographische Replikation relevanten verschiedenen Regionen einer Cloud, auch auf über mehrere Verfügbarkeitszonen verteilt werden [145]. Dies kann aber je nach Einstellung zu einer erhöhten Latenz durch die zusätzlich benötigte Synchronisation führen.

Data Store Type
Charakteristisch für die geographische Replikation ist die Verteilung der Daten, damit ist hier nur der distributed Data Store Type relevant.

4.3.5 Datenverteilung (Sharding)
Bezüglich der Cloud Data Hosting Solutions gibt es bei der Datenverteilung keine Unterschiede zum reinen Outsourcing des DBL.

4.3.6 Off-Loading of Peak Loads (Cloud Burst)
Data Store Type
Charakteristisch für die Cloud Burst ist die temporäre Auslagerung der Daten um sehr große und meist nicht vorhersagbare Lasten zu bedienen. Damit werden i.d.R. zumindest Lese-Replikas benötigt, womit nur der Distributed Data Store Type relevant ist.

4.3.7 Arbeiten auf Datenkopie (Datenanalyse und Monitoring)
Bezüglich der Cloud Data Hosting Solutions gibt es beim Arbeiten auf Datenkopien keine Unterschiede zum reinen Outsourcing des DBL.

4.3.8 Datensynchronisation
Bezüglich der Cloud Data Hosting Solutions gibt es bei der Datensynchronisation keine Unterschiede zum reinen Outsourcing des DBL.

4.3.9 Backup
Service Model
Für Backups sind auch SaaS Service Modelle verfügbar. So kann z.B. der Online Speicherdienst Dropbox oder spezielle Cloud Backup-Lösungen wie Datacastle genutzt werden um Dateien in der Cloud zu sichern.

4.3.10 Archivierung
Bezüglich der Cloud Data Hosting Solutions gibt es bei der Archivierung keine Unterschiede zum Backup.

4.3.11 Datenimport aus der Cloud
Service Model
Für den Datenimport aus der Cloud werden vor allem SaaS Service Modelle genutzt. So kann z.B. die API des Mikroblogging Dienstes Twitter genutzt werden um Social Media Monitoring zu betreiben. Aber auch PaaS und IaaS Service Modelle sind denkbar wenn Dritten Zugriff auf Datenbankdienste oder einen Rohdatenspeicherbereich gewährt wird.

Data Store Type
Aus Sicht des Konsumenten der Daten aus der Cloud importiert spielt es keine Rolle ob die Daten im Hintergrund centralized oder distributed gehalten werden. Daher spielt dieses Kriterium für das Szenario Datenimport aus der Cloud keine Rolle.

Compatibility
Beim Datenimport aus der Cloud werden die Daten über eine anbieterspezifische API abgefragt. Somit sind die Rückgabewerte in einem anbieterspezifischen Format und müssen häufig vor der Speicherung in das eigene Datenformat umgewandelt werden.

4.4 Abbildung der Migrationsszenarien auf Cloud Data Patterns

4.4.1 Überblick Cloud Data Patterns

Durch Migration des DBL in die Cloud stehen ggf. Funktionen des ursprünglichen Quellsystems nicht mehr zur Verfügung oder die Daten dürfen auf Grund ihrer Vertraulichkeit nicht 1:1 in die Cloud gelangen. Dazu hat Strauch et al. in [12] und [4] Cloud Data Patterns identifiziert, die diesen Umständen gerecht werden. Die Patterns sind unterteilt in Functional Patterns (*Data Store Functionality Extension* und *Emulator of Stored Procedures*), Non-Functional Patterns (*Local Database Proxy* und *Local Sharding-Based Router*) und Confidentiality Patterns (*Confidentiality Level Data Aggregator, Confidentiality Level Data Splitter, Filter of Critical Data, Pseudonymizer of Critical Data* und *Anonymizer of Critical Data*).

Die Functional Patterns erweitern Cloud Datenspeicher um Funktionalität, die im Quellsystem vorhanden war, aber im Cloud Datenspeicher fehlt. Das Data Store Functionality Extension Pattern ermöglicht z.B. Joins, die in einigen NoSQL Datenbanken nicht zur Verfügung stehen. Dabei soll die zusätzliche Funktionalität durch einen Adapter in der Cloud emuliert werden und durch die Nähe zum Cloud Datenspeicher die Latenz gering halten. Das Emulator of Stored Procedures Pattern ist ein Spezialfall des Data Store Functionality Extension Pattern für Stored Procedures. Hier wird im DAL anstatt eines Stored Procedure Aufrufs die entsprechende Simulation der Stored Procedure im Adapter aufgerufen und das Ergebnis zurückgegeben.

Die Non-Functional Patterns stellen QoS Attribute sicher und sorgen hauptsächlich für kurze Latenzen durch Lese-Replikate in Verbindung mit einem lokalen Proxy innerhalb des DAL (Local Database Proxy) oder durch einen lokalen Proxy der Sharding für Nicht-Sharding-fähige Cloud Datenspeicher ermöglicht (Local Sharding-Based Router).

Die Confidentiality Patterns bieten verschiedene Möglichkeiten um zu verhindern, dass kritische Daten z.B. in öffentliche Clouds gelangen. Sie können z.B. gefiltert (Filter of Critical Data), anonymisiert (Anonymizer of Critical Data) oder pseudonymisiert (Pseudonymizer of Critical Data) werden. Außerdem ist es möglich bei verschiedenen Vertraulichkeitsstufen sicherzustellen, dass bei nicht-atomaren Daten (in Bezug auf verschiedene Vertraulichkeitsstufen) die höchste Vertraulichkeitsstufe für die gesamten Daten gilt (Confidentiality Level Data Aggregator). Daten können auch so aufgeteilt werden, dass sie je nach Vertraulichkeit in einem passenden Cloud Datenspeicher abgelegt werden (Confidentiality Level Data Splitter).

Cloud Data Pattern / Szenario	Reines Outsourcing des DBL	Hochskalierbare Datenspeicher	Geographische Replikation	Sharding	Cloud Burst	Arbeiten auf Datenkopie	Datensynchronisation	Backup	Archivierung	Datenimport aus der Cloud	Datennutzung aus der Cloud
Functional Patterns											
Data Store Functionality Extension	x	x	x	x	x	x	x	-	-	-	x
Emulator of Stored Procedures	x	x	x	x	x	x	x	-	-	-	x
Non-Functional Patterns											
Local Database Proxy	x	x	x	x	x	-	x	-	-	-	-
Local Sharding-Based Router	x	x	x	x	x	-	x	-	-	-	-
Confidentiality Patterns											
Confidentiality Level Data Aggregator	x	x	x	x	x	x	x	-	-	(x)	x
Confidentiality Level Data Splitter	x	x	x	x	x	x	x	-	-	(x)	x
Filter of Critical Data	x	x	x	x	x	x	x	-	-	(x)	x
Pseudonymizer of Critical Data	x	x	x	x	x	x	x	-	x	(x)	x
Anonymizer of Critical Data	x	x	x	x	x	x	x	-	-	(x)	x

Tabelle 3 Abbildung der Cloud Data Patterns auf Cloud Datenmigrationsszenarien

Tabelle 3 zeigt welche Cloud Data Patterns für welche Cloud Datenmigrationsszenarien in Frage kommen (mit „x“ markiert) und welche untypisch oder nicht sinnvoll sind (mit „-“ markiert).

Um Wiederholungen zu vermeiden werden im folgenden Abschnitt die Abbildungen der Cloud Data Patterns auf das reine Outsourcing des DBL begründet und für die weiteren Cloud Datenmigrationsszenarien nur noch die Abweichungen zum reinen Outsourcing des DBL bezüglich der Abbildung der Patterns beschrieben.

4.4.2 Reines Outsourcing des DBL

Data Store Functionality Extension

Da beim reinen Outsourcing des DBL zwar kein Wechsel des Datenspeichertyps, aber des Produkts vorkommen kann, müssen teilweise produktspezifische Funktionen wie spezielle Datentypen durch einen Adapter nachgebildet werden.

Emulator of Stored Procedures

Nicht alle Cloud Datenspeicher unterstützen Stored Procedures. SQL Azure z.B. unterstützt Stored Procedures nicht im gleichen Umfang wie Microsoft SQL Server, möchte man sie dennoch im vollen Umfang benutzen, benötigt man für die fehlenden Funktionen einen Emulator.

Local Database Proxy
Um die Zugriffszeit auf Cloud Datenspeicher zu verringern, können Lese-Replikate des Masters in der Cloud angelegt und ein lokaler Proxy in den DAL integriert werden, um Leseanfragen transparent an ein Lese-Replikat weiterzuleiten.

Local Sharding-Based Router
Um weiter horizontale Skalierbarkeit zu ermöglichen, können Daten auch auf verschiedene Knoten verteilt werden (Sharding). Damit dies auch für Cloud Datenspeicher möglich ist, die kein Sharding unterstützen und für die Anwendungsschicht transparent für abläuft, kann ein Local Sharding-Based Router im DAL eingesetzt werden, welcher die Daten aus den entsprechenden Shards lädt bzw. speichert.

Confidentiality Level Data Aggregator
Sofern ein reines Outsourcing des DBL in eine Public Cloud vorgesehen ist, dürfen meist kritische Daten nicht in den Public Cloud Datenspeicher gelangen. Um zu entscheiden welche Daten aus verschiedenen Quellsystemen kritisch sind, kann ein Confidentiality Level Data Aggregator eingesetzt werden, welcher auch die möglicherweise verschiedenen Vertraulichkeitsstufen angleicht.

Confidentiality Level Data Splitter
Beim reinen Outsourcing des DBL kann es notwendig sein, Daten gemäß ihrer Vertraulichkeitsstufe in unterschiedliche Cloud Datenspeicher zu schreiben bzw. aus ihnen zu lesen. Ein Confidentiality Level Data Splitter im DAL kann dies transparent für die Anwendungsschicht übernehmen.

Filter of Critical Data
Um zu verhindern, dass kritische Daten z.B. in eine Public Cloud gelangen, kann ein Filter of Critical Data im DAL eingesetzt werden. Dieser filtert transparent für die Anwendungssicht kritische Daten heraus und speichert diese z.B. nicht in einem Public Cloud Datenspeicher.

Pseudonymizer of Critical Data
Wenn es ausreichend ist, mit pseudonymisierten Daten im Cloud Datenspeicher zu arbeiten, kann ein Pseudonymizer of Critical Data im DAL dafür sorgen, dass kritische Daten automatisch pseudonymisiert werden und somit nicht z.B. in eine Public Cloud gelangen.

Anonymizer of Critical Data
Wenn es ausreichend ist, mit anonymisierten Daten im Cloud Datenspeicher zu arbeiten, kann ein Anonymizer of Critical Data im DAL dafür sorgen, dass kritische Daten automatisch anonymisiert werden und somit nicht z.B. in eine Public Cloud gelangen.

Untergruppe Datennutzung aus der Cloud
Die Non-Functional Patterns Local Database Proxy und Local Sharding-Based Router spielen für die Datennutzung aus der Cloud keine Rolle, da es hier nicht auf geringe Latenzen ankommt, wie in Abschnitt 4.1.2 beschrieben.

4.4.3 Verwendung von Hochskalierbaren Datenspeichern (NoSQL, BLOB Datenspeicher)

Bezüglich der Cloud Data Patterns gibt es bei der Verwendung von hochskalierbaren Datenspeichern keine Unterschiede zum reinen Outsourcing des DBL.

4.4.4 Geographische Replikation

Bezüglich der Cloud Data Patterns gibt es bei der geographischen Replikation keine Unterschiede zum reinen Outsourcing des DBL.

4.4.5 Datenverteilung (Sharding)

Bezüglich der Cloud Data Patterns gibt es bei der Datenverteilung keine Unterschiede zum reinen Outsourcing des DBL.

4.4.6 Off-Loading of Peak Loads (Cloud Burst)

Bezüglich der Cloud Data Patterns gibt es beim Off-Loading of Peak keine Unterschiede zum reinen Outsourcing des DBL.

4.4.7 Arbeiten auf Datenkopie (Datenanalyse und Monitoring)

Local Database Proxy

Für das Arbeiten auf einer Datenkopie spielt der Local Database Proxy keine Rolle, da hier typischerweise das gleiche Datenspeicher-Produkt eingesetzt wird.

Local Sharding-Based Router

Für das Arbeiten auf einer Datenkopie spielt der Local Sharding-Based Router keine Rolle, da hier typischerweise das gleiche Datenspeicher-Produkt eingesetzt wird.

4.4.8 Datensynchronisation

Bezüglich der Cloud Data Patterns gibt es bei der Datensynchronisation keine Unterschiede zum reinen Outsourcing des DBL.

4.4.9 Backup

Da es sich beim Backup um eine reine Kopie des DBL handelt, auf der nicht weitergearbeitet wird, sondern nur im Falle eines Ausfalls möglichst schnell in die Datenschicht importiert wird, spielen die Cloud Data Patterns hier keine Rolle.

4.4.10 Archivierung

Pseudonymizer of Critical Data

Bei der Archivierung kann ein Pseudonymizer of Critical Data genutzt werden um auch kritische Daten in pseudonymisierter Form in einer Public Cloud zu sichern. Die Abbildungen zwischen den Pseudonymen und Echtdaten müssen aber weiter in einem sicheren Speicher gehalten werden um eine Wiederherstellung der Originaldaten zu gewährleisten.

Die weiteren Cloud Data Patterns spielen bei der Archivierung keine Rolle, da hier nicht direkt auf den archivierten Daten weitergearbeitet wird, sondern nur im Falle einer Analyse oder für Beweise das Archiv in die Datenschicht importiert wird.

4.4.11 Datenimport aus der Cloud

Da es sich beim Datenimport aus der Cloud um eine Nutzung einer API handelt, auf deren Datenspeicher man in der Regel keinen Einfluss hat, spielen die Functional und Non-Functional Cloud Data Patterns hier keine Rolle. Sofern die zu importierenden Daten vertraulich sind, können die Confidentiality Patterns helfen die passende Vertraulichkeitsstufe zu finden, den vertraulichen Teil der Daten in einen sicheren Datenspeicher abzulegen oder vor der Speicherung herauszufiltern, zu anonymisieren oder zu pseudonymisieren.

4.5 Datenarten bei der Migration

Neben den eigentlichen Nutz- oder Anwendungsdaten spielen bei der Cloud Datenmigration noch andere Arten von Daten eine Rolle. Diese werden in diesem Abschnitt näher betrachtet.

4.5.1 Anwendungsdaten

Die Anwendungs- oder auch Nutzdaten müssen bei der Cloud Datenmigration möglichst vollständig vom Quell- in das Zielsystem migriert werden. Dazu stehen verschiedene Möglichkeiten wie die Nutzung von Import-Tools, ETL-Tools oder dem simplen Kopieren von Rohdaten zur Verfügung. Zu Beginn sollte eine Menge an repräsentative Daten ausgewählt und die Migration dieser Daten getestet werden. Häufige Umformungen sind erfahrungsgemäß bei Datums- und Zeitangaben, sowie der Darstellung von Booleschen Werten notwendig. Bei NoSQL Datenbanken ist es zudem häufig notwendig zusammengesetzte Primärschlüssel in einfache Primärschlüssel zu konvertieren, da der Zugriff auf die Objekte einer NoSQL Datenbank i.d.R. nur über einen Schlüssel möglich ist.

Für Performancesteigerungen und bessere Skalierbarkeit sollten Daten die momentan in RDBMS gehalten werden, nicht Bestandteil von Transaktionen sind und häufig gelesen oder geschrieben werden in NoSQL Dienste ausgelagert werden [35].

Um die Performance bei der Migration von großen Datenmengen zu erhöhen, sollte das Exportieren und Importieren parallelisiert und sonstige I/O-lastige Anwendungen auf dem Quell- und Zielsystem pausiert werden. Weiterhin kann die Indexierung während der Migration deaktiviert werden und nach dem Import der Daten einmal vollständig durchgeführt werden. Dadurch kann der Import insgesamt schneller durchgeführt werden. [17] Sofern der Cloud Anbieter es unterstützt, können große Datenmengen auch per Postweg zum Anbieter geschickt werden [146], oder von einem Technologiepartner mit schnellerer und direkter Anbindung in ein Rechenzentrum des Cloud Anbieters übertragen werden [43].

Einige NoSQL Datenbanken unterstützen nur eine Sortierung in lexikographischer Reihenfolge. In diesem Fall muss sichergestellt werden, dass (negative) Zahlen korrekt sortiert werden, was durch Zero-Padding und einen Offset mindestens in Höhe des Betrags der maximale vorstellbaren negativen Zahl erreicht werden kann. [97]

Weiter gibt es bei einigen NoSQL Datenbanken Größenbeschränkungen für Attribute, womit sich keine größeren Texte oder Binärdaten direkt speichern lassen, sondern nur Referenzen auf Objekte in einem BLOB Datenspeicher gespeichert werden können. [98]

4.5.2 Schemata, Views und Indexe

Die Schema-Migration spielt nur eine Rolle, falls das Quell- und Zielsystem ein RDBMS ist, da bei NoSQL- oder BLOB-Speichern i.d.R. keine Schemata existieren. Sofern es sich bei dem Produkt, welches ein RDBMS zur Datenspeicherung nutzt, um Standardsoftware handelt, gibt es ggf. vom Hersteller Migrationstools, die dabei helfen Daten inkl. Schemata etc. von einem RDBMS Produkt zum anderen zu migrieren [17].

Bei der Migration von einem RDBMS auf einen NoSQL Speicher geht das feste Schema verloren und die Gruppierung der Daten muss auf den entsprechenden NoSQL Speicher angepasst werden. So werden häufig zusammengehörige Daten in einem Datensatz (verschachtelt) gespeichert, anstatt sie gemäß der dritten und folgenden Normalformen auf verschiedene Entitäten aufzuteilen. Dabei ist zu beachten dass die Speichergröße

pro Datensatz häufig beschränkt ist und somit ein Kompromiss zwischen zusammengehörigen Daten und Datensatzgröße gefunden werden muss. Dieser Kompromiss muss so ausgelegt sein, dass auch beim Anwachsen der verschachtelten Attribute die Datensatzgröße nicht überschritten wird.

Wenn es sich bei der Migration von einem RDBMS in ein anderes RDBMS nicht um die gleiche Version oder das gleiche Produkt handelt, kann es unter Umständen notwendig sein, auch das Ziel-Schema anzupassen. Dazu gehören z.B. Anpassungen oder Ersetzung bestimmter Datentypen inkl. Kollation und Präzision (Dezimalzahlen, Time Stamps), Standardwerte, sortierter Speicherung, Partitionierung, Namen und Bezeichner auf Grund von Kollisionen oder Einschränkungen (Sonderzeichen, reservierte Worte) und Werte-Einschränkungen (Constraints), die nicht in allen Produkten existieren oder anders benannt sind. Weiter bieten einige Datenspeicher die Möglichkeit Zeitstempel für bestimmte Spalten automatisch bei jeder Änderung zu setzen, welche ggf. durch Trigger, den DAL oder höhere Schichten simuliert werden müssen. [17]

Auch bei Indexen spielt die Kollation, Groß- und Kleinschreibung für Vergleiche und Sortierung eine Rolle, z.B. wenn bei Abfragen die Groß- und Kleinschreibung (Case Sensitivity) berücksichtigt werden soll. [17]

Steht eine Funktion im Ziel-RDBMS nicht zur Verfügung, muss sie wenn möglich als Trigger simuliert werden [17] oder von einer höheren Schicht, z.B. dem DAL, übernommen werden (siehe Data Store Functionality Extension Pattern in Abschnitt 4.4.1). Wenn der DAL eine Funktionalität simuliert, muss diese möglichst nah am Datenspeicher ausgeführt werden, um Latenzprobleme zu verringern.

Zum Schema gehört meist auch die Definition von Indexen, welche oft wesentlich mehr Speicher als die Anwendungsdaten selbst benötigen, aber dadurch schnellere Lesezugriffe auf bestimmte Daten ermöglichen. Dieser zusätzliche Speicherbedarf muss mit in Betracht gezogen werden, wenn es Größeneinschränkungen in Cloud RDBMS Diensten, wie Amazon RDS gibt.

Die Migration von Views ist häufig ohne großen Aufwand möglich, da sie nur aus SQL Befehlen bestehen, ansonsten gelten dieselben Besonderheiten wie bei der Schema-Migration. [17]

4.5.3 Trigger und Stored Procedures

Trigger und Stored Procedures enthalten Anwendungslogik, die nah an den Daten ausgeführt werden soll, um große Datentransfers zur Anwendungsebene zu vermeiden. Diese Anwendungslogik ist aufwändig zu migrieren, da sie teilweise in proprietären Programmiersprachen beschrieben ist oder von Hersteller zu Hersteller unterschiedliche Funktionen bietet (z.B. bietet Oracle keine Transaktionsunterstützung in Triggern). Tests sollten sicherstellen, dass die Semantik bei der Migration erhalten bleibt. [17]

4.5.4 Administrationsskripte

Datenbankadministratoren (DBA) verwalten RDBMS in der Regel mit Administrationsskripten z.B. für Performance Monitoring, Nutzer Wartung, Objekt Wartung oder DB Wartung. Diese müssen an das neue Zielsystem, sofern sie noch benötigt werden, angepasst oder können ggf. verworfen werden. [17]

4.5.5 Konfigurationsdaten

Quellsysteme sind häufig vom DBA so konfiguriert, dass typische Anfragen möglichst schnell beantwortet werden. Dazu muss der DBA Wissen über die Häufigkeit und Art der Anfragen haben und damit das Quellsystem möglichst passend daraufhin einzustellen. Einfache Beispiele sind hier die vom Quellsystem zu verwendende Arbeitsspeicher- und Cachegröße oder maximale Anzahl paralleler Threads. All diese Konfigurationen sollten, falls möglich, auch in das Zielsystem übertragen und dabei ggf. angepasst werden um keine Performanceprobleme zu bekommen. [17]

Weitere Konfigurationen wie Benutzer, Gruppen oder Rollen und Rechte müssen auch im Zielsystem eingerichtet werden. [17]

Cloud Datenspeicher lassen sich nicht immer im gleichen Umfang konfigurieren, wie die lokalen Quellsysteme, dies betrifft vor allem PaaS und SaaS Lösungen. IaaS Lösungen können Software-seitig meist vollständig wie lokale Quellsysteme konfiguriert werden, allerdings ist die Auswahl an passender Hardware meist eingeschränkt.

4.5.6 Mandantenfähigkeit

Die vorherigen Unterabschnitte treffen auch auf Datenmigrationen außerhalb der Cloud zu, besonders wichtig für die Cloud Datenmigration ist jedoch die Mandantenfähigkeit, weshalb sie hier separat behandelt wird.

Um die Kosten-, Skalierbarkeits- und Elastizitätseigenschaften der Cloud weiter ausnutzen zu können, können die Daten aller Mandanten einer Anwendung die zuvor in separaten Datenspeichern gehalten wurden, in der Cloud in einen mandantenfähigen Datenspeicher migriert werden. [147]

Dazu können verschiedene Anpassungen an den Anwendungsdaten, als auch am Schema und damit auch an allen anderen Datenarten notwendig sein. So können die Anwendungsdaten statt in verschiedenen Datenspeicherinstanzen (Shared Nothing/Shared HW) in einer Instanz (Shared DB/Separate Schema/Shared Schema) gespeichert werden [147]. Dabei muss beachtet werden, dass keine Schlüssel doppelt vorkommen, was z.B. durch die Verwendung von Universally Unique Identifier (UUIDs), Globally Unique Identifier (GUIDs) oder einen Mandantenpräfix sichergestellt werden kann. Bei UUIDs und GUIDs kann es zusätzlich notwendig sein, pro Datensatz ein Attribut zu speichern, welches den Datensatz einem Mandanten (Mandantenschlüssel) zuordnet um z.B. alle Datensätze eines bestimmten Mandanten abzufragen. Der Mandantenschlüssel ist ein zusätzlich eingeführter Schlüssel der wiederum selbst durch ein UUID, GUID oder auch eine User ID oder Tenant ID dargestellt werden kann.

Die Migration aller Anwendungsdaten von allen Mandanten in eine Datenspeicherinstanz (Shared Schema) hat häufig Auswirkungen auf höhere Schichten. Der DAL kann hier die Isolation der Daten zusammen mit Nutzerrechten und Views bei SQL fähigen Datenspeichern übernehmen und somit nur Anfragen bezüglich eines Mandanten akzeptieren, wenn der entsprechende Mandantenschlüssel übereinstimmt [147]. So kann der DAL Anfragen auf die ursprüngliche Tabelle auf den View des Mandanten umleiten oder im einfacheren Fall, je nach Schlüsselstrategie, bei jeder Abfrage einen Filter auf den Mandantenschlüssel setzen [148]. Die Anwendungsschicht muss je nach Schlüsselstrategie und DAL dennoch ggf. angepasst werden, z.B. wenn der DAL nicht mächtig genug ist bei jeder Operation auf den Daten den Mandantenschlüssel als Filterkriterium zu

setzen oder ein PaaS Dienst genutzt wird, der einen bestimmten DAL vorschreibt (wie Google App Engine [149]) bzw. keine Views erlaubt.

4.6 Auswirkungen der Migration auf Datenzugriffs- und Anwendungsschicht

Bei der Migration der Datenschicht in die Cloud müssen je nach Migrationsszenario und verwendetem Produkt oder Dienst auch Anpassungen an höheren Schichten vorgenommen werden. Diese wurden in den vorherigen Abschnitten bereits beschrieben und werden in den folgenden Abschnitten zusammengefasst.

4.6.1 Überblick Auswirkungen der Migration

Die folgende Tabelle zeigt im Überblick welche Auswirkungen in welchen Migrationsszenarien auftreten können. Anschließend werden die einzelnen Auswirkungen näher beschrieben.

Auswirkung / Szenario	**Reines Outsourcing des DBL**	**Hochskalierbare Datenspeicher**	**Geographische Replikation**	**Sharding**	**Cloud Burst**	**Backup**	**Archivierung**	**Arbeiten auf Datenkopie**	**Datensynchronisation**	**Datenimport aus der Cloud**	**Datennutzung aus der Cloud**
Netzwerkebene											
CNAME	x	x	x	x	x	x	x	x	x	-	x
DNS Cache TTL	x	x	x	x	x	x	x	x	x	-	x
Portfreigabe (Firewall)	x	x	x	x	x	x	x	x	x	-	x
Data Access Layer											
Verbindungskennung	x	x	x	x	x	x	x	x	x	-	x
Cloud Data Pattern	x	x	x	x	x	x	x	x	x	-	x
Synchronisation	-	x	x	x	x	x	x	x	x	-	-
Cloud Burst Migration	-	-	-	-	x	-	-	-	-	-	-
Semantik (Schema/DB-Name)	x	x	x	x	x	-	-	x	x	-	x
Namenskonventionen	x	x	x	x	x	-	-	x	x	-	x
Datentypen	x	x	x	x	x	-	-	x	x	-	x
Semantik („“ vs. Null)	x	x	x	x	x	-	-	x	x	-	x
Sortierung	x	x	x	x	x	x	x	x	x	x	x
Attributgrößen	x	x	x	x	x	x	x	x	x	x	x
Anwendungsebene											
Datenspeichertypwechsel	-	x	x	x	x	x	x	x	x	x	-
Komplexe Anfragen Outsourcen	-	x	-	-	-	-	-	-	-	-	-
Cloud Data Pattern Auswirkung	x	x	x	x	x	x	x	x	x	x	x
Produktwechsel (Datentyp, etc.)	x	x	x	x	x	x	x	x	x	x	x
Semantik (Vergleiche, Locks)	x	x	x	x	x	-	-	x	x	-	x

Tabelle 4 Abbildung der Anpassungen pro Anwendungsschicht auf Cloud Datenmigrationsszenarien

Auf eine Begründung der Abbildung der Auswirkungen auf die Migrationsszenarien wird hier verzichtet, da sie im Wesentlichen aus den Abschnitten 4.1 bis 4.4, sowie den folgenden Beschreibungen hervorgeht.

Die Anpassungen pro Schicht und Szenario können hier nicht vollständig ermittelt werden, da sie auf den konkreten Gegebenheiten der Ausgangssituation basieren, sie enthalten jedoch Anhaltspunkte für gängige Anpassungen.

4.6.2 Anpassungen auf Netzwerkebene

Die einfachste Anpassung ist die Änderung des *CNAME* Eintrags im DNS Server. Sofern eine URL für die Adressierung des Datenspeichers verwendet wird und das Quell- und Zielsystem kompatibel sind, reicht es den CNAME Eintrag auf die IP Adresse des neuen Zielsystems umzustellen. Weiter ist sicherzustellen, dass die Systeme, die das neue Zielsystem benutzen sollen eine kurze Verfallszeit (Time to Live, *TTL*) für ihre lokalen *DNS Cache* Einträge haben.

Damit die ggf. lokalen Anwendungen auf einen Cloud Datenspeicher zugreifen können, müssen i.d.R. zusätzlich *Firewalls* neu konfiguriert werden um Verbindungen zwischen dem lokalen Netzwerk und Cloud Datenspeicher zu ermöglichen. Eventuell ist es auch sinnvoll eine virtuelle private Cloud (VPC [150]) einzurichten.

Neben dem Produktivsystem des Datenspeichers müssen auch, falls vorhanden, die Monitorskripte des Produktivsystems angepasst werden, um im Falle eines Ausfalls des Produktivsystems den CNAME Eintrag des DNS Servers automatisch auf die Backup- bzw. Failover-Instanz umzuleiten.

4.6.3 Anpassungen auf DAL Ebene

Alternativ zur Änderung des CNAME Eintrags im DNS Server kann auch die *Verbindungskennung* im DAL auf das neue Zielsystem angepasst werden. Dies ist ausreichend, wenn das Quell- und Zielsystem kompatibel ist. Ansonsten müssen zumindest Adapter dafür sorgen, dass nicht unterstützte Funktionen oder Konfigurationen des neuen Zielsystems simuliert werden. Dies ist mit den *Cloud Data Patterns* die in Abschnitt 4.4.1 beschrieben wurden möglich.

Bei der geographischen Replikation muss der DAL und darauf aufbauende Schichten damit umgehen können, dass die gleichen Schichten parallel auch in anderen Rechenzentren laufen können und im Hintergrund der DBL *synchronisiert* wird.

Im *Cloud Burst* Szenario kann z.B. der DAL entscheiden, wann es notwendig ist die Daten temporär in eine Cloud bzw. zurück in das lokale Rechenzentrum zu migrieren. Diese Entscheidung kann aber auch von externen Monitoring Tools getroffen werden, welche z.B. entsprechende Skripte aufrufen können um die Daten zu migrieren.

Im Datensynchronisationsszenario muss die *Synchronisation* der lokalen Replik mit der Master Datenbank in der Cloud eingerichtet werden, diese Funktionalität kann z.B. von einigen RDBMS selbst übernommen werden, oder muss z.B. durch den DAL oder externe Synchronisations-Tools realisiert werden.

Bei einigen Datenspeichern ist die *Semantik von Schema und Datenbankname* verschieden (z.B. Oracle vs. Microsoft SQL Server) und kann vom DAL konvertiert werden. Außerdem sind die *Namenskonventionen* von Datenspeichern unterschiedlich, generell

sollten deshalb kurze, großgeschriebene Namen ohne Sonderzeichen und potentiell reservierten Worten verwendet werden. Der DAL kann hier in bestimmten Fällen inkompatible Namenskonventionen zwischen Quell- und Zieldatenspeicher für die Anwendungsebene transparent auflösen. Gleiches gilt für *Datentypen* die nicht von allen Zielspeichern unterstützt werden, hier kann der DAL z.B. automatisch eine Umwandlung vornehmen, wie von BOOLEAN zu BIT oder CHAR. [17]

Problematisch ist die Umwandlung bei *semantisch* nicht eindeutigen Konvertierungen, wie z.B. der Vergleich eines Datums gegen einen *leeren Wert* („“) oder gegen *keinen Wert* (Null). Oracle z.B. unterstützt keine leeren Werte, hier wird in der Regel eine Anpassung auf Anwendungsebene notwendig, um sicherzustellen, dass die Semantik der Abfragen nicht verletzt wird. [17]

Bei einigen NoSQL Speichern, wie Amazon SimpleDB, geschieht eine *Sortierung* der Ergebnismenge, sofern diese Funktion überhaupt unterstützt wird, nur in lexikographischer Reihenfolge. Daher muss insbesondere bei (negativen) Zahlen darauf geachtet werden, dass ein Offset (mindestens der Betrag der maximal vorstellbaren negativen Zahl) und Zero-Padding verwendet wird um korrekt sortierte Ergebnisse zu erhalten. Der DAL kann den Offset hinzu- bzw. herausrechnen, bevor die Daten gespeichert, bzw. an höhere Schichten weitergegeben werden. [97]

Weiter sind die *Attributgrößen* bei NoSQL Speichern zum Teil stark eingeschränkt und erlauben keine Speicherung von längeren Texten oder größeren Binärdaten (z.B. über 1 kB [98]). Hier wird empfohlen die größeren Werte in einem BLOB Datenspeicher zu hinterlegen und als Attribut nur einen Pointer auf das Objekt im BLOB Datenspeicher zu verwenden. [9]

4.6.4 Anpassungen auf Anwendungsebene

Bei der *Änderung des Datenspeichertyps* vom Quell- zum Zielsystem ist es häufig nicht ausreichend fehlenden Funktionen im DAL zu emulieren. Wenn z.B. eine Migration von einem RDBMS auf einen NoSQL oder BLOB Datenspeicher stattfindet und ggf. keine definierten Schemata, Joins innerhalb von Abfragen, strikte Konsistenz oder Transaktionen mehr unterstützt werden, muss die Anwendungsebene an diese konzeptionellen Änderungen angepasst werden.

Dies kann bedeuten, dass für bestimmte Anwendungsfälle zusätzlich zu einem NoSQL oder BLOB Datenspeicher noch ein RDBMS benötigt wird, da ein transaktionales Verhalten für diese Anwendungsfälle geschäftskritisch ist, aber für die meisten Anwendungsfälle ein NoSQL oder BLOB Datenspeicher ausreicht. (siehe Abschnitte 2.1.2 und 2.4)

Außerdem kann es nützlich sein, *externe Dienste* wie Amazon MapReduce zu verwenden um *komplexe Anfragen* inkl. Joins auf NoSQL Datenspeichern wie Amazon DynamoDB auszuführen anstatt diese in der Anwendungsebene zu programmatisch durchzuführen, was zu einer höheren Latenz und Traffic führen kann. [151]

Bei gleichem Quell- und Ziel-Datenspeichertyp aber einem Produktwechsel kann es zu *unterschiedlichen Semantiken* z.B. der *Vergleichsoperatoren* kommen, welche in der Anwendungsebene beachtet werden müssen. Des Weiteren ist bei der Verwendung von herstellerabhängigen SQL Befehlen zu prüfen, ob diese auch im Zielsystem unterstützt werden. [17]

Sofern im DAL *Cloud Data Patterns* zur Vertraulichkeit verwendet wurden, kann es sein, dass die Anwendungsebene keine volle Sicht mehr auf die Daten hat und muss ggf. daraufhin angepasst werden.

Bei RDBMS existieren trotz einiger Standardisierungen wie der Abfrage- und Manipulationssprache SQL noch *semantische Unterschiede* z.B. im *Lock*-Verhalten, die in einigen Produkten z.B. dirty reads erlauben oder Locking in unterschiedlichen Granularitätsstufen verwenden. Diese Unterschiede müssen vor der Migration identifiziert werden und die Anwendungsebene daraufhin angepasst werden. [17]

4.7 Methodik

In diesem Abschnitt werden die Ergebnisse der vorherigen Abschnitte zusammengeführt und eine gesamtheitliche Cloud Datenmigrationsmethodik entwickelt.

4.7.1 Überblick Cloud Datenmigrationsmethodik

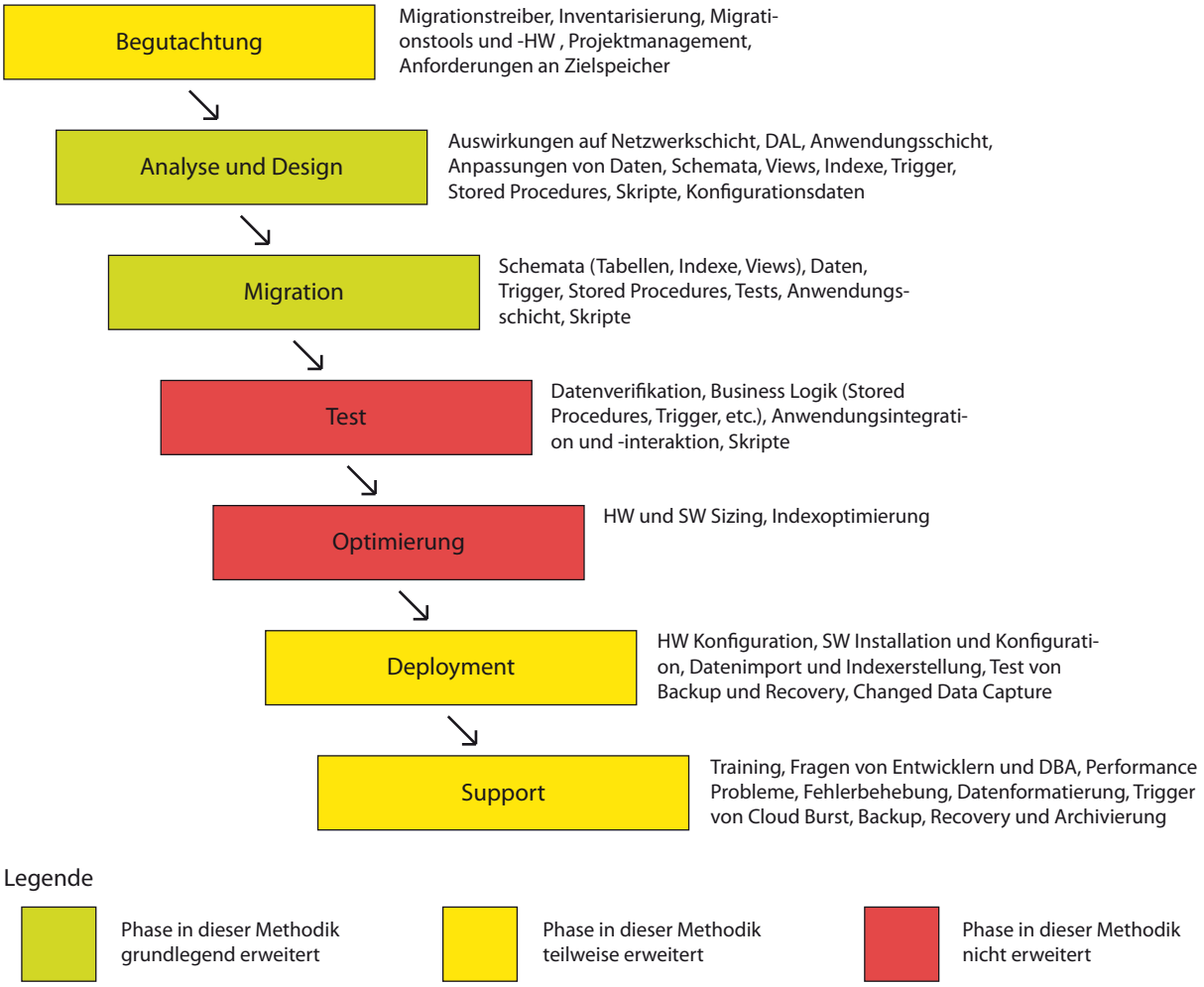

Abbildung 28 Cloud Datenmigrationsmethodik nach Laszewski [17]

Laszewski beschreibt in [17] bereits eine Methodik für Migrationsprojekte, die aus dem klassischen Wasserfallmodell abgeleitet wurde. Sie beinhaltet die Phasen Assessment, Analysis and Design, Migration, Testing, Optimization, Deployment und Post-Product Support. Die Ausführungen zu den einzelnen Phasen beinhalten allerdings nur das RDBMS von Oracle als Zieldatenspeicher, sowie SQL Server, Sybase, DB2 oder Informix als relationale Quellsysteme. Abbildung 28 zeigt eine Adaption von Laszewskis Migrati-

ons-Projektlebenszyklus mit den wichtigsten Schritten innerhalb jeder Phase und einer Kennzeichnung welche Phasen in dieser Arbeit erweitert werden.

Die in dieser Arbeit vorgestellte Migrationsmethodik beinhaltet neben dem teilweise auch in [17] vorgestellten reinen Outsourcing des DBL neun zusätzliche Migrationsszenarien (siehe Abschnitt 4.1) und zwei weitere Quell- und Zieldatenspeichertypen (NoSQL und BLOB Datenspeicher). Weiterhin ist diese Migrationsmethodik erweiterbar z.B. um Aspekte wie Security oder Compliance, die in den einzelnen Phasen als Schritte hinzugefügt werden können.

Die hier gezeigte Cloud Datenmigrationsmethodik kann sowohl eigenständig, als auch innerhalb einer Cloud Anwendungsmigration, wie z.B. von Amazon in [9] und Abbildung 3 gezeigt, verwendet werden. Dabei enthält die hier geschilderte Begutachtungsphase sämtliche, für die Datenmigration relevanten Schritte für Amazons Cloud Assessment Phase. Die Analyse und Design Phase enthält relevante Schritte zur Datenmigration für Amazons Proof of Concept und dem ersten Teil der Data Migration Phase. Die Migrationsphase entspricht dem letzten Teil von Amazons Data Migration Phase und Teilen der Application Migration Phase, was die Auswirkungen der Datenmigration auf höhere Schichten anbelangt. Die Analyse und Design, sowie Migrationsphase enthält weiterhin die für die Datenmigration relevanten Schritte von Amazons Leverage the Cloud Phase. Eine explizite Test, Deployment und Post-Production Support Phase gibt es in Amazons Phasenmodell nicht und die Optimierungsphase enthält alle, für die Datenmigration relevanten Schritte der Optimization Phase.

Die Begutachtungsphase erweitert die Methodik dieser Arbeit um die Identifikation des Migrationsszenarios inkl. der unterschiedlichen Ausprägungen jedes Szenarios. Die Analyse und Design Phase wird im Inhalt um Kriterien erweitert, die für von RDBMS verschiedenen Zieldatenspeichertypen relevant sind. Sie beinhaltet zusätzlich Cloud Data Hosting Solutions, Vorschläge zur Anpassung höherer Anwendungsschichten und Cloud Data Patterns, welche auch in der Migrationsphase Anwendung finden für ein möglicherweise notwendiges Refactoring. Die Test- und Optimierungsphase wird auf die zusätzlichen Szenarien angewandt, aber sonst in dieser Arbeit nicht erweitert, da sie in [17] generisch genug für alle Zieldatenspeichertypen ist. Ein Cloud Datenmigrationstool hilft in prototypischen Szenarien in der Analyse und Design-, Migrations- und Deploymentphase. Die abschließende Supportphase wird in dieser Arbeit um Schritte erweitert, die in bestimmten Szenarien nach dem eigentlichen Einrichten der Migration wiederholt werden. Dies ist besonders für das Cloud Burst, Backup und Archivierungsszenario relevant, da hier in den Phasen Begutachtung bis Deployment der Cloud Burst-, Backup- und Archivierungsprozess angelegt und getestet, jedoch nicht unbedingt sofort im Produktionssystem durchgeführt, sondern in der Zukunft regelmäßig wiederholt wird.

Migrationsphase / Szenario	**Reines Outsourcing des DBL**	**Hochskalierbare Datenspeicher**	**Geographische Replikation**	**Sharding**	**Cloud Burst**	**Arbeiten auf Datenkopie**	**Datensynchronisation**	**Backup**	**Archivierung**	**Datenimport aus der Cloud**	**Datennutzung aus der Cloud**
Begutachtung											
Migrationstreiber	x	x	x	x	x	x	x	x	x	x	x
Inventarisierung	x	x	x	x	x	(x)	x	(x)	x	x	x
Anforderungen Zielspeicher	x	x	x	x	x	x	x	x	x	x	x
Migrationstools und -HW	x	x	x	x	x	x	x	x	x	x	x
Migrationsszenario	x	x	x	x	x	x	x	x	x	x	x
Projektmanagement	x	x	x	x	x	x	x	x	x	x	x
Analyse und Design											
Auswahl des Zielspeichers	x	x	x	x	x	(x)	x	x	x	x	x
Datenbereinigung	x	x	x	x	x	x	x	x	x	x	x
Planung Änderung Datenarten	x	x	x	x	x	(x)	x	x[3]	x	x	x
Planung Änderung höherer Schichten	x	x	x	x	x	-	x	-	x	x	x
Planung Änderung Systemland-schaft	x	x	x	x	x	(x)	x	-	x	x	x
Migration											
Datenarten	x	x	x	x	x	(x)	x	x	x	x	x
Höhere Schichten	x	x	x	x	x	-	x	-	x	x	x
Systemlandschaft	x	x	x	x	x	(x)	x	-	x	x	x
Test											
Test des Systems	x	x	x	x	x	x	x	x	x	x	x
Korrektur des Systems	x	x	x	x	x	x	x	x	x	x	x
Test des Gesamtsystems	x	x	x	x	x	(x)	x	-	x	x	x
Korrektur des Gesamtsystems	x	x	x	x	x	(x)	x	-	x	x	x
Optimierung											
Installation und Simulation	x	x	x[4]	x	x	-	x	-	x	x	x
Optimierung	x	x	x[4]	x	x	-	x	-	x	x	x
Test nach Optimierung	x	x	x[4]	x	x	-	x	-	x	x	x
Deployment											
HW Konfiguration	x	x	x	x	x	x	x	x	x	x	x
SW Installation und Konfigurati-	x	x	x	x	x	x	x	x	x	x	x

[3] Nur Backup und Recovery Skripte als Datenarten spielen hier eine Rolle

[4] Bei der Nutzung eines CDNs für statische Daten spielt die Optimierungsphase keine Rolle, bei der gesamten geographischen Verteilung jedoch schon.

on											
Datenimport und Indexerstellung	x	x	x	x	x[5]	x	x	-	x[6]	x	x
Test Backup und Recovery	x	x	x	x	x	(x)	x	x	x	x	x
Übertragung der Änderungen	x	x	x	x	-	-	x	-	-	x	x
Post-Production Support											
Training, Support, Fehlerbehebung	x	x	x	x	x	x	x	x	x	x	x
Cloud Burst - Trigger und Migrationen	-	-	-	-	x	-	-	-	-	-	-
Backup - Trigger und Erstellung	-	-	-	-	-	-	-	x	-	-	-
Recovery - Trigger und Rückspielung	-	-	-	-	-	-	-	x	-	-	-
Archivierung - Trigger und Durchführung	-	-	-	-	-	-	-	-	x	-	-

Tabelle 5 Abbildung der Schritte einzelner Migrationsphasen auf Cloud Datenmigrationsszenarien

Tabelle 5 zeigt eine Übersicht, welche Schritte der einzelnen Migrationsphasen für die jeweiligen Migrationsszenarien eine Rolle spielen. Eine notwendige Durchführung eines Schritts im jeweiligen Szenario ist mit „x" gekennzeichnet und für eine unmögliche oder nicht sinnvolle Ausprägung im jeweiligen Szenario steht ein „-". Ein „(x)" kennzeichnet eine mögliche, aber untypische Ausprägung.

Im folgenden Abschnitt wird die Cloud Datenmigrationsmethodik für das Szenario „Reines Outsourcing des DBL" beschrieben. In den darauf folgenden Abschnitten wird die Methodik für alle weiteren Szenarien in Form von Abweichungen zum Vorgehen beim Reinen Outsourcing des DBL erläutert.

4.7.2 Reines Outsourcing des DBL

Die hier vorgestellte Methode kann in die Migration des Gesamtsystems eingebettet werden, wenn die Migration des DBL ein Teil der Migration des Gesamtsystems ist, aber die hier in Abschnitt 4.7 eingeführte Methode adressiert ausschließlich die Migration der Datenbankschicht inklusive erforderlichen Anpassungen der anderen Anwendungsschichten.

Für alle Phasen gelten die vier goldenen Regeln der Datenmigration von Morris [56]. Diese verbessern die Wirtschaftlichkeit einer Datenmigration indem Perfektionismus vermieden wird und es eine enge Abstimmung mit dem Business gibt, welche Daten in welcher Qualität migriert werden müssen.

Begutachtung

Die Begutachtungsphase von Laszewski [17] wird im Folgenden kurz zusammengefasst und ergänzt um In- und Outputdaten, sowie die Identifikation des Migrationsszenarios (Schritt Fünf). In der Begutachtungsphase werden gemäß einem Migrationsprojekt keine neuen Anforderungen gesammelt, sondern Informationen für das Projektmanagement gesammelt, eine Kostenschätzung erstellt, Herangehensweise ausgewählt, nutzbare und relevante Tools identifiziert und das Anwendungsportfolio inventarisiert

[5] Der Datenimport und die Indexerstellung spielt beim Cloud Burst Szenario eine besondere Rolle, siehe Abschnitt 4.7.6

[6] Bei Archivierungen werden i.d.R. keine Indexe benötigt, da diese Daten nicht regelmäßig genutzt werden.

um Auswirkungen der Migration für andere IT Systeme und Prozesse (Anwendungen, Integrationsdienste, Reporting, Backup, Recovery) zu ermitteln.

Eintrittsbedingungen der Begutachtungsphase

- Entscheidung, dass eine Datenmigration stattfinden soll

Austrittsbedingungen der Begutachtungsphase

- Migrationstreiber (Auslöser und Gründe für die Migration) identifiziert
- Auswirkung der Migration auf Systemlandschaft ermittelt
- Anforderungen an Zielspeicher ermittelt
- Migrationstools und benötigte Hard- und Software während der Migration identifiziert
- Migrationsszenario mit spezifischer Ausprägung identifiziert
- Projektplan erstellt

Schritte innerhalb der Begutachtungsphase

1. Identifikation der Migrationstreiber [17]
    a. *Inputdaten*: Auftrag, Interview mit Auftraggeber und weiteren Stake Holdern
    b. *Outputdaten*: priorisierte Liste von Migrationstreibern
2. Inventarisierung der Systemlandschaft [17]
    a. *Inputdaten*: Dokumentationen der Systemlandschaft (Systemarchitekturplan), Spezifikationen, Interview mit Systemarchitekt
    - *Outputdaten*: Übersicht der Systemlandschaft auf welcher alle Auswirkungen der Migration identifiziert werden können
        - Anwendungen die mit dem Quellspeicher interagieren – lesen via SQL oder API
        - Anwendungen die direkt Transkationen auf dem Quellspeicher ausführen – Anwendungen, Stored Procedures oder SQL Manipulationen
        - Anwendungen oder Skripte die Daten im- oder exportieren
        - Management- oder Administrationsskripte
        - externe Interfaces
3. Ermittlung der Anforderungen an den Zielspeicher [17]
    a. *Inputdaten*: Auftrag, Beziehungen des Quellsystems zu Drittsystemen
    b. *Outputdaten*: Anforderungen an den Zielspeicher
4. Identifikation von möglichen Migrationstools und -HW [17]
    a. *Inputdaten*: Herstellerangaben des Quell- und Zielspeichers, Internet-Recherche, Auftrag und Anforderungen an Zielspeicher (für HW Bedarf während der Migration)
    b. *Outputdaten*: Liste an Migrationstools und benötigter HW während der Migration
5. Identifikation des Migrationsszenarios mit spezifischen Ausprägungen (siehe Abschnitt 4.2)
    a. *Inputdaten*: Beschreibung der Migrationsszenarien, Ausprägungen gemäß Taxonomie (Ablösung, Verbleibende lokale Existenz des DBL, Grad der Migration, Quell-Datenspeichertyp, Ziel-Datenspeichertyp, Versions-/Produktwechsel des DBL, Richtung der Datenbewegung, Dauerhaftigkeit der Migration des DBL, Zeitspanne der Migration, Variabilität der Res-

sourcennutzung des DBL, Anstieg der Ressourcenauslastung des DBL) (siehe Abschnitt 4.2.1)
   b. *Outputdaten*: Migrationsszenario mit spezifischen Ausprägungen
6. Projektmanagement [17]
   a. *Inputdaten*: Auftrag, verfügbare personelle Ressourcen und Qualifikationen, Aufwandsschätzungen
   b. *Outputdaten*: Projektplan

Analyse und Design Phase
In der Analyse und Design Phase werden die Implementierungsdetails identifiziert und beschrieben [17]. Dazu werden die in Abschnitt 4.3 beschriebenen Kriterien der Cloud Data Hosting Solutions, die in Abschnitt 4.5 aufgezeigten Anpassungen der Datenarten und die in Abschnitt 4.6 beschriebenen Auswirkungen auf andere Anwendungsschichten herangezogen. Die Cloud Data Patterns, welche wiederverwendbare Lösungen für mögliche Probleme bei der Datenmigration beschreiben (Abschnitt 4.4) dienten dabei als Vorlage für Abschnitt 4.6.

Eintrittsbedingungen der Analyse und Design Phase
- Auswirkung der Migration auf Systemlandschaft ermittelt
- Anforderungen an Zielspeicher ermittelt
- Migrationstools und benötigte Hard- und Software während der Migration identifiziert
- Migrationsszenario mit spezifischer Ausprägung identifiziert

Austrittsbedingungen der Analyse und Design Phase
- Auswahl des Zielspeichers (Typ, Anbieter, Produkt/Dienstleistung, Version) festgelegt
- Detaillierte Pläne zur Umsetzung der Änderungen, um den Betrieb nach der Migration fehlerfrei aufrecht zu halten, an:
  - den verschiedenen Datenarten (inkl. Bereinigung der Anwendungsdaten),
  - höheren Anwendungsschichten und
  - der bestehenden Systemlandschaft

Schritte innerhalb der Analyse und Design Phase
1. Auswahl des Zielspeichers und Registrierung bei potentiellen Cloud Data Store Providern zur Verifikation der Anforderungen
   a. *Inputdaten*: Anforderungen an Zielspeicher, Cloud Data Hosting Solutions, Anbieterliste mit Produktbeschreibungen
   b. *Outputdaten*: Zielspeicher (Typ, Anbieter, Produkt/Dienstleistung, Version)
2. Planung der Datenbereinigung [55]
   a. Inputdaten: Anwendungsdaten, Qualitätsansprüche an die Anwendungsdaten (bzgl. Dubletten, Vollständigkeit, etc.)
   b. Outputdaten: Maßnahmen (z.B. Skripte) zur Bereinigung der Anwendungsdaten
3. Planung der Änderungen an den verschiedenen Datenarten [17]
   a. *Inputdaten*: Datenarten die von der Migration betroffen sind, Liste von Inkompatibilitäten zwischen Quell- und Zielspeicher
   b. *Outputdaten*: Auflistung der benötigten Änderungen pro Datenart, ggf. als ETL Prozess oder Migrationsskript [45]

4. Identifikation der Auswirkungen und Planung der Änderungen auf höhere Anwendungsschichten [17]
    a. *Inputdaten*: vorliegendes Migrationsszenario, Systementwurf mit Beschreibung des DAL und der Anwendungsschicht, ggf. auch Quellcode
    b. *Outputdaten*: Auflistung der benötigten Anpassungen pro Schicht
5. Planung der Änderungen an der Systemlandschaft [17]
    a. *Inputdaten*: Auswirkung der Migration auf Systemlandschaft
    b. *Outputdaten*: Auflistung der benötigten Anpassung pro System

Migrationsphase
Die Migrationsphase beschäftigt sich mit der eigentlichen Umsetzung der in der Analyse und Design Phase erstellten Pläne.

Eintrittsbedingungen der Migrationsphase
- Plan der Änderungen an den Datenarten erstellt und geprüft
- Plan der Änderungen am System (pro Schicht) erstellt und geprüft
- Plan der Änderungen an angrenzenden Systemen (pro System) erstellt und geprüft
- Zielspeicher ermittelt

Austrittsbedingungen der Migrationsphase
- Migration der Datenarten abgeschlossen
- Anpassung der höheren Schichten abgeschlossen
- Anpassung der angrenzenden Systeme abgeschlossen

Schritte innerhalb der Migrationsphase
1. Migration der Datenarten [17]
    a. *Inputdaten*: Auflistung der benötigten Änderungen pro Datenart, Anwendungsdaten, Datendefinitionen (Schemata, Views, Indexe), Quellcode der Trigger und Stored Procedures, Skripte, Konfigurationsdaten
    b. *Outputdaten*: migrierte Anwendungsdaten, Datendefinitionen (Schemata), Trigger, Stored Procedures, Skripte und Konfigurationsdaten
2. Anpassung der höheren Schichten [17]
    a. *Inputdaten*: Auflistung der benötigten Änderungen pro Schicht, Quellcode des Systems
    b. *Outputdaten*: angepasster Quellcode des Systems
3. Anpassung der angrenzenden Systeme [17]
    a. *Inputdaten*: Auflistung der Anpassungen pro System, Quellcode (inkl. Konfigurationsdateien) der anzupassenden Systeme
    b. *Outputdaten*: angepasster Quellcode (inkl. Konfigurationen) der angrenzenden System

Testphase
Nach der eigentlichen Migration und den Anpassungen muss deren Erfolg intensiv funktional getestet werden. Sofern Fehler identifiziert werden, müssen sie in dieser Phase behoben und die Systeme erneut getestet werden.

Eintrittsbedingungen der Testphase
- Angepasstes und ausführbares System mit migrierten Datenarten
- Angepasste und ausführbare angrenzende Systeme

Austrittsbedingungen der Testphase

- Alle durch den Test aufgedeckten Fehler und Probleme wurden behoben (funktional)

Schritte innerhalb der Testphase

1. Test des Systems
    a. *Inputdaten*: Angepasstes und ausführbares System mit migrierten Datenarten, Testfälle (für Datenverifikation – keine Fehler während Migration, z.B. gleiche Spaltenlängen, Business-Logik Test, Skript Test [17])
    b. *Outputdaten*: bestandene und fehlgeschlagene Testfälle
2. Korrektur des Systems (sofern fehlgeschlagene Testfälle existieren)
    a. *Inputdaten*: Angepasstes und ausführbares System mit migrierten Datenarten, Testfälle (insbesondere fehlgeschlagene Testfälle)
    b. *Outputdaten*: Dokumentation, dass alle Testfälle bestanden wurden (falls nicht, Wiederholung des Schritts Zwei)
3. Test der angrenzenden Systeme
    a. *Inputdaten*: Angepasste und ausführbare angrenzende Systeme, Testfälle (für Anwendungsintegration und -interaktion)
    b. *Outputdaten*: bestandene und fehlgeschlagene Testfälle
4. Korrektur der angrenzenden Systeme (sofern fehlgeschlagene Testfälle existieren)
    a. *Inputdaten*: Angepasste und ausführbare angrenzende Systeme, Testfälle (insbesondere fehlgeschlagene Testfälle)
    b. *Outputdaten*: Dokumentation, dass alle Testfälle bestanden wurden (falls nicht, Wiederholung des Schritts Vier)

Optimierungsphase

Noch vor dem Deployment wird das migrierte System für einen längeren Zeitraum mit typischen Abfragen konfrontiert um die nicht-funktionalen Eigenschaften zu testen und ggf. Performanceprobleme zu identifizieren und zu beheben. [17]

Eintrittsbedingungen der Optimierungsphase

- Funktional getestetes Gesamtsystem mit den behobenen Fehlern und Problemen aus der Testphase

Austrittsbedingungen der Optimierungsphase

- unter Performance-Gesichtspunkten (Verfügbarkeit, Zuverlässigkeit, Skalierbarkeit) optimiertes System (nicht-funktional)

Schritte innerhalb der Optimierungsphase

1. Installation des angepassten und migrierten Systems auf die Zielinfrastruktur (oder eine ihr ähnlichen Infrastruktur) und Simulation des Betriebs
    a. *Inputdaten*: Angepasstes, migriertes und ausführbares System, Muster typischer Abfragen, Monitoring-Werkzeuge
    b. *Outputdaten*: Performanceprobleme
2. Optimierung des Systems, ggf. durch HW Sizing oder Indexoptimierung [17]
    a. *Inputdaten*: Performanceprobleme, Monitoring-Berichte, Schemata, aktuelle Infrastruktur Spezifikation
    b. *Outputdaten*: optimierte Indexe, optimierte Infrastruktur Konfiguration
3. Test auf optimierten Indexen und Infrastruktur

a. *Inputdaten*: optimiertes System, Muster typischer Abfragen, Monitoring-Werkzeuge
b. *Outputdaten*: Dokumentation, dass alle Performanceprobleme bestanden wurden (falls nicht, Wiederholung ab Schritt Zwei)

Deployment
Nachdem sichergestellt ist, dass alle funktionalen (Testphase) und nicht-funktionalen (Optimierungsphase) Anforderungen erfüllt werden, kann das Gesamtsystem in der Deploymentphase produktionsreif gemacht und in Betrieb genommen werden. [17]

Eintrittsbedingungen der Deploymentphase
- Funktional und nicht-funktional voll funktionsfähiges und getestetes System

Austrittsbedingungen der Deploymentphase
- in Betrieb genommenes, vollständig migriertes Gesamtsystem (migrierter DBL, angepasstes Gesamtsystem entsprechend Schritt 3.3 und 3.4 der Analyse und Design Phase)

Schritte innerhalb der Deploymentphase
1. HW Konfiguration (Speicher, Netzwerk, Cluster, Recovery) [17]
    a. *Inputdaten*: HW, Konfigurationsbeschreibung
    b. *Outputdaten*: konfigurierte HW
2. SW Installation und Konfiguration (Schema, Nutzer, Rechte, Gruppen, Verbindungskennungen) [17]
    a. *Inputdaten*: SW, Installationsanleitungen, Konfigurationsbeschreibung, Beschreibung der Schemata, Views, Trigger, Stored Procedures
    b. *Outputdaten*: installierte und konfigurierte SW
3. Datenimport und Indexerstellung [17]
    a. *Inputdaten*: Anwendungsdaten, Migrationsskripte, Indexdefinitionen
    b. *Outputdaten*: migrierte Daten inkl. zugehöriger Indexe
4. Test der Backup und Recovery Skripte und Prozesse [17]
    a. *Inputdaten*: Backup und Recovery Skripte
    b. *Outputdaten*: Dokumentation, dass der Backup- und Recovery-Prozess funktionsfähig ist
5. Übertragen der Änderungen (Changed Data Capture) [17]
    a. *Inputdaten*: Changed Data, Migrationsskripte
    b. *Outputdaten*: vollständig migriertes und produktiv eingesetztes System

Post-Production Support
In der abschließenden Post-Production Support Phase stehen Personen, die die Migration durchgeführt haben noch für einen bestimmten Zeitraum nach dem Deployment für Trainings und Fragen der Entwickler oder DBAs zur Verfügung. Sie beheben im Produktivbetrieb auftretende Fehler im Hinblick auf funktionale sowie nicht-funktionale Anforderungen. Zum Beispiel bzgl. Performance Problemen, fehlender oder falscher Funktionalität, oder korrigieren ggf. Datenformatierungen die in vorherigen Tests nicht aufgedeckt wurden. [17]

Untergruppe Datennutzung aus der Cloud
Bezüglich der einzelnen Schritte der Cloud Datenmigrationsmethodik gibt es bei der Datennutzung aus der Cloud keine Unterschiede zum reinen Outsourcing des DBL.

4.7.3 Verwendung von Hochskalierbaren Datenspeichern (NoSQL, BLOB Datenspeicher)

Bezüglich der einzelnen Schritte der Cloud Datenmigrationsmethodik gibt es bei der Verwendung von hochskalierbaren Datenspeichern keine Unterschiede zum reinen Outsourcing des DBL.

4.7.4 Geographische Replikation

Im Falle einer bloßen Verteilung von statischen Daten über ein CDN müssen keine Daten migriert, sondern nur kopiert werden und die Referenzen auf die ursprünglichen Daten ggf. angepasst werden. Somit ist eine komplexe Migration von Datenarten nicht notwendig, sondern lediglich eine Kopie zu erstellen bzw. ein bestimmter Daten Container als Quelle für die Verteilung festzulegen. Dennoch müssen höhere Schichten und die Systemlandschaft ggf. angepasst werden, um auf die verteilten Daten zuzugreifen, womit auch die Begutachtungs-, Analyse und Design, Test, Deployment und Post-Production Support Phase notwendig sind. Die Optimierungsphase bei der Nutzung eines CDN für statische Daten fällt hier weg.

Soll eine ganze Anwendung geographisch verteilt werden, gibt es bezüglich der durchzuführenden Schritte keine Unterschiede zum reinen Outsourcing des DBL. Es sollte aber der Synchronisationsaspekt hervorgehoben werden, der beim reinen Outsourcing noch meist regional beschränkt war und weniger Performanceprobleme mit sich brachte.

4.7.5 Datenverteilung (Sharding)

Neben den im Folgenden beschriebenen Ergänzungen in der Analyse und Design Phase gibt es bei der Datenverteilung (Sharding) bezüglich der einzelnen Schritte der Cloud Datenmigrationsmethodik keine Unterschiede zum reinen Outsourcing des DBL.

Analyse und Design

In den Schritten 3 bis 5 der Analyse und Design Phase muss zusätzlich festgelegt werden, wie die Daten auf unterschiedliche Shards verteilt werden sollen (siehe Abschnitt 4.1.5), sofern dies nicht automatisch vom Zielspeichersystem übernommen wird [90] [85].

4.7.6 Off-Loading of Peak Loads (Cloud Burst)

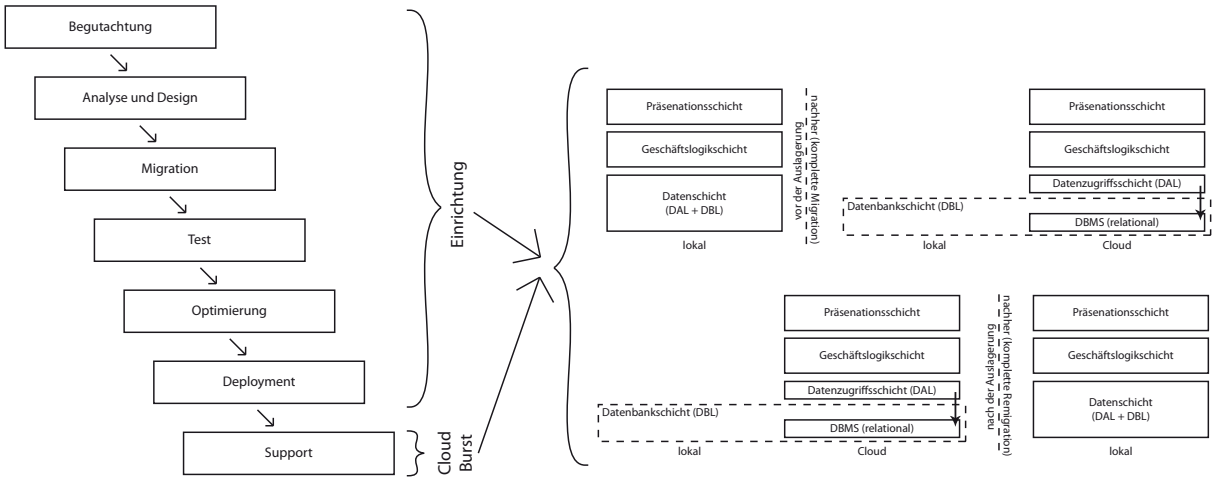

Abbildung 29 Zuordnung der Migrationsphasen zu den Migrationen im Cloud Burst Szenario

Im Cloud Burst Szenario existieren, wie Abbildung 29 dargestellt, pro Cloud Burst Fall zwei Migrationen. Zuerst müssen die Daten in die Cloud und anschließend wieder zurück in den ursprünglichen Datenspeicher migriert werden. Die gesamte Cloud Daten-

migrationsmethodik bezieht sich dabei auf das Einrichten und Testen einer Cloud Burst Funktionalität (Phasen Begutachtung bis Deployment), die dann im Falle von hoher Last regelmäßig genutzt wird (Phase Post-Production Support).

Deployment
In dieser Phase werden beim Cloud Burst Szenario alle Vorbereitungen getroffen, damit bei hoher Last eine spontane Datenmigration in die Cloud und bei sinkender Last eine Remigration aus der Cloud geschehen kann. Dabei wird auch Schritt Drei (Datenimport und Indexerstellung) einmal für die Migration und einmal für die Remigration exemplarisch ausgeführt und im Schritt Vier die Backup und Recovery Prozesse auch im Cloud Burst Fall getestet. Schritt Fünf (Übertragung der Änderungen) hingegen existiert nicht, da Schritt Drei offline getestet wird und somit alle Daten bereits migriert sind (siehe unten).

Post-Production Support
Schritt Drei der Deploymentphase (Datenimport und Indexerstellung) wird pro Cloud Burst Fall automatisch oder manuell angestoßen und läuft danach automatisiert ab, dies ist im zweiten Schritt (Cloud Burst Trigger und Migrationen) der Post-Production Support Phase festgehalten. Die Beendigung des Cloud Burst Falls wird ebenfalls automatisch oder manuell angestoßen und läuft dann für die Remigration automatisch ab (wieder Schritt Drei der Deploymentphase mit Datenimport und Indexerstellung). Somit müssen zwei automatisierte Versionen von Schritt Drei der Deploymentphase existieren, eine für die Migration und eine für die Remigration.

4.7.7 Arbeiten auf Datenkopie (Datenanalyse und Monitoring)

Begutachtung, Analyse und Design, Migration und Test
Bei dem Arbeiten auf einer Datenkopie fällt Schritt Zwei der Begutachtungsphase (Inventarisierung) leichtgewichtiger aus, da Datenkopien selten Auswirkungen auf höhere Anwendungsschichten und externe Systeme haben, es sei denn ein externes System hat zuvor mit den Produktivdaten gearbeitet und soll nun mit der Datenkopie arbeiten. Somit spielen auch die Schritte bezüglich höherer Schichten in den Phasen Analyse und Design, Migration und Test keine Rolle und die Schritte bezüglich der gesamten Systemlandschaft nur, wenn ein bestehendes System zukünftig auf der Datenkopie arbeiten soll.

Schritt Eins, Zwei und Drei der Analyse und Design Phase (Auswahl des Zielspeichers, Datenbereinigung und Planung Änderung Datenarten), sowie Schritt Eins der Migration (Datenarten) ist meist trivial, da wie in Abschnitt 4.1.7 beschrieben, die gleiche Produktversion verwendet wird und lediglich eine passende HW gewählt werden muss und die Daten aus dem Produktivsystem exportiert und in das Zielsystem importiert werden.

Optimierung
Die gesamte Optimierungsphase ist für das Arbeiten auf Datenkopien nicht notwendig, da hier Daten lediglich kopiert und nicht migriert werden.

Deployment
Schritt Vier des Deployments (Test Backup und Recovery) ist im Arbeiten auf Datenkopien Szenario meist irrelevant, da schon vom Quellsystems Backups erstellt werden. Es kann allerdings sinnvoll sein die Ergebnisse, die aus der Analyse der Datenkopien entstanden sind, zu sichern und ggf. wiederherzustellen. Da es sich beim Arbeiten auf Da-

tenkopien, wie bei einem Backup, um einen konsistenten Snapshot handelt, gibt es auch keinen Schritt Fünf (Übertragung der Änderungen).

4.7.8 Datensynchronisation

Bezüglich der einzelnen Schritte der Cloud Datenmigrationsmethodik gibt es bei der Datensynchronisation keine Unterschiede zum reinen Outsourcing des DBL.

4.7.9 Backup

Die gesamte Cloud Datenmigrationsmethodik bezieht sich beim Backup auf das Einrichten und Testen einer Backup Funktionalität, die dann manuell oder automatisch ausgelöst und danach automatisiert ablaufen kann.

Begutachtung, Analyse und Design, Migration und Test

Beim Backup fällt Schritt Zwei der Begutachtungsphase (Inventarisierung) leichtgewichtiger aus, da Backups meist nur lokale Auswirkungen haben und somit für höhere Anwendungsschichten, als auch externe Systeme keine oder kaum Einfluss haben. Somit spielen auch die Schritte bezüglich höherer Schichten und der Systemlandschaft in den Phasen Analyse und Design, Migration und Test keine Rolle.

Optimierung

Die Optimierungsphase ist im Backup Szenario dafür verantwortlich, sicherzustellen, dass der Produktivbetrieb während der Erstellung eines Backups, z.B. durch erhöhte Latenzen oder kurze Nicht-Verfügbarkeit während der Erstellung eines Snapshots, nicht gestört wird.

Deployment

Schritt Zwei des Deployments (Datenimport und Indexerstellung) ist im Backup Szenario irrelevant, da i.d.R. Backups in ein quellgleiches System oder als Export in Form von Binärdaten gespeichert werden und somit Daten kopiert und nicht importiert oder migriert werden. Da es sich beim Backup um einen konsistenten Snapshot handelt, gibt es auch keinen Schritt fünf (Übertragung der Änderungen). Sollte es sich um inkrementelle Backups handeln, zählt dies nicht unter Schritt Fünf, sondern ist ein erneuter Durchlauf des Backup-Prozesses.

Post-Production Support

Der Backup-Teil von Schritt Vier der Deploymentphase wird im Backup Szenario automatisch oder manuell angestoßen und läuft danach automatisiert ab, dies ist im dritten Schritt (Backup Trigger und Erstellung) der Post-Production Support Phase festgehalten. Muss ein Backup in Produktivsystem zurückgespielt werden (Recovery), wird der Recovery-Teil von Schritt Vier der Deploymentphase ausgeführt, welcher im vierten Schritt (Recovery Trigger und Rückspielung) der Post-Production Support Phase festgehalten ist.

4.7.10 Archivierung

Deployment

Da es sich bei der Archivierung meist um einen Teil der Daten handelt, die zwar alt, aber in sich konsistent sind, gibt es auch keinen Schritt Fünf (Übertragung der Änderungen).

Post-Production Support

Der Datenimport-Teil von Schritt Drei der Deploymentphase wird im Archivierungsszenario automatisch oder manuell angestoßen und läuft danach automatisiert ab, dies ist

im fünften Schritt (Archivierung Trigger und Durchführung) der Post-Production Support Phase festgehalten.

4.7.11 Datenimport aus der Cloud

Bezüglich der einzelnen Schritte der Cloud Datenmigrationsmethodik gibt es beim Datenimport aus der Cloud keine Unterschiede zum reinen Outsourcing des DBL.

4.8 Fragebögen

Die in diesem Abschnitt erstellten Fragebögen dienen der Identifizierung des vorliegenden Migrationsszenarios mit jeweils ihrer genauen Ausprägung, der ausgewählten Cloud Data Solution und benötigter Cloud Data Patterns. Basierend auf den entwickelten Fragebögen kann im Cloud Datenmigrations-Tool die Anbieterauswahl eingeschränkt und Konflikte identifiziert werden.

4.8.1 Szenario Identifikation

Ein Cloud Datenmigration kann mehrere der in Abschnitt 4.1 aufgezeigten Szenarien umfassen. Der Fragebogen zur Szenario Identifikation hat demnach zum einen das Ziel die Szenarien zu identifizieren, zum anderen die Ausprägungen pro Szenario zu identifizieren und später mögliche Konflikte aufzuzeigen (siehe Abschnitt 5.3.3).

Dazu werden im ersten Schritt die beinhalteten Szenarien des Migrationsprojektes und im zweiten Schritt die spezifischen Ausprägungen identifiziert.

Schritt Eins: Szenario Identifikation

Welche Szenarien treffen auf Ihr Migrationsprojekt zu? (Mehrfachauswahl möglich)

☐ Reines Outsourcing des Data Base Layer (DBL)
Outsourcing des DBL bedeutet die Migration des lokalen DBL in die Cloud ohne dabei den Typ des Datenspeichers (RDBMS, NoSQL oder BLOB Datenspeicher) zu verändern.

☐ Verwendung von Hochskalierbaren Datenspeichern (NoSQL, BLOB Datenspeicher)
Bei der Verwendung von hochskalierbaren Datenspeichern wird davon ausgegangen, dass zuvor ein RDBMS benutzt wurde, welches aber nicht hochskalierbar ist und die Daten nun in einen hochskalierbaren Cloud Datenspeicher migriert werden soll.

☐ Geographische Replikation
Um die Latenz beim Zugriff auf Daten möglichst gering zu halten werden die Daten zum einen möglichst nah an den Ort wo die Datenverarbeitung stattfindet (data shipping) und/oder zum anderen möglichst nah an den Benutzer gebracht. Dabei kann es bei berechneten Daten auch sinnvoll sein die Berechnungen nah an die Daten (function shipping) und damit auch nah an den Benutzer zu bringen, was einer Replikation der Anwendungslogik entspricht. Umfassend kann dies mit der Replikation der Datenschicht und der dazugehörigen höheren Anwendungsschichten in ein dem Benutzern naheliegendes Cloud Rechenzentrum ermöglicht werden.

☐ Datenverteilung (Sharding)
Die Daten werden disjunkt auf verschiedene Rechenzentren verteilt. Auch dieser Ansatz kann Daten näher an Benutzer bringen und somit die Latenzen verringern. Außerdem ermöglicht dieser Ansatz eine bessere Skalierbarkeit, da die Daten nun über mehrere DB-Instanzen oder DB-Cluster verteilt werden.

☐ Off-Loading of Peak Loads (Cloud Burst)
Off-Loading of Peak Loads, auch Cloud Burst genannt, ist die temporäre Nutzung von Ressourcen in der Cloud und somit die temporäre Auslagerung des gesamten oder Teilen des DBL. Ursache dafür ist ein zeitlich begrenzter sehr hoher Ressourcenbedarf, der nicht oder nicht kosteneffizient mit den Ressourcen des lokalen Rechenzentrums bewerkstelligt werden kann. Dazu werden entweder Ressourcen aus der Cloud hinzugeschaltet und übernehmen somit die zusätzliche Last oder die gesamte Infrastruktur einer Anwendung wird temporär in die Cloud ausgelagert und läuft dort so lange, bis die Ressourcen im lokalen Rechenzentrum wieder ausreichen um die Last wieder selbst zu bewältigen.

☐ Arbeiten auf Datenkopie (Datenanalyse und Monitoring)
Beim Arbeiten auf einer Datenkopie wird eine reine Kopie des gesamten oder teilweisen DBL erstellt. Diese Kopie dient in der Regel der Entlastung von Produktivsystemen, wenn auf den gleichen Daten komplexe Analysen erstellt werden sollen.

☐ Datensynchronisation
Die Datensynchronisation mit der Cloud ermöglicht es mehreren Nutzern auf relativ aktuellen Daten offline zu arbeiten. Dazu wird der bisher lokale DBL in die Cloud kopiert und danach eine Synchronisation eingerichtet. Damit ermöglicht man zeitweise offline zu gehen und trotzdem ein relativ aktuelles Bild des Gesamtdatenbestandes zu haben.

☐ Backup
Ein Backup beinhaltet die reine Kopie des DBL z.B. in die Cloud, es hält einen Zustand zu einem bestimmten Zeitpunkt fest und dient der Sicherheit um bei Ausfällen die Daten wiederherzustellen.

☐ Archivierung
Archivierung ist ähnlich dem Backup auch eine reine Kopie des gesamten oder teilweisen DBL, erfüllt aber einen anderen Zweck. Zum einen müssen aus regulatorischen Gründen bestimmte Daten für eine bestimmte Zeit aufbewahrt werden und zum anderen können Daten, die im Produktivsystem nicht mehr benötigt werden, entfernt werden um mehr Platz für neue Daten zu schaffen. Eine Wiederherstellung von archivierten Daten ist nicht für den Fehlerfall gedacht, sondern zum einen für Analysen und zum anderen für Beweise z.B. in gerichtlichen Prozessen.

☐ Datenimport aus der Cloud
Einige Cloud Dienste stellen ihren Daten über eine API zum Abruf zur Verfügung. Damit können externe Daten aus Cloud Diensten abgerufen und lokal

verarbeitet sowie gespeichert werden. Hierbei wird also der gesamte oder Teile des DBL eines Cloud Dienstes in das lokale Rechenzentrum importiert um mit geringer Latenz auf den Daten zu arbeiten.

☐ Datennutzung aus der Cloud
Bei der Datennutzung aus der Cloud wird der DBL einer Anwendung in die Cloud migriert und die lokale Anwendung greift zukünftig nicht mehr auf den lokalen DBL zu, sondern auf den entfernten DBL in der Cloud. Die damit einhergehende höhere Latenz kann für bestimmte Anwendungen, z.B. selten genutzte Anwendungen einer Abteilung, vernachlässigt werden.

Schritt Zwei: Identifikation der spezifischen Szenario Kriterien

Welche Ablösestrategie verfolgt die Migration? (Einfachauswahl)
☐ Live (ohne Downtime) ☐ Non-Live (mit Downtime)

Was soll mit dem lokalen DBL geschehen? (Mehrfachauswahl möglich)
☐ Verschieben des DBL in die Cloud
☐ Kopie des DBL in die Cloud
☐ Einweg-Synchronisation
☐ Zweiweg-Synchronisation

Welchen Grad hat die Migration? (Einfachauswahl)
☐ vollständig ☐ teilweise

Um welche Quell-Datenspeichertypen handelt es sich? (Mehrfachauswahl möglich)
☐ RDBMS ☐ NoSQL ☐ BLOB

Um welche Ziel-Datenspeichertypen handelt es sich? (Mehrfachauswahl möglich)
☐ RDBMS ☐ NoSQL ☐ BLOB

Gibt es einen Versions- oder Produktwechsel des DBL bei der Migration? (Einfachauswahl)
☐ gleiches Produkt, unterschiedliche Version
☐ gleiches Produkt, gleiche Version
☐ unterschiedliches Produkt

In welche Richtung erfolgt die Datenbewegung bei der Migration? (Einfachauswahl)
☐ von lokalem Rechenzentrum in die Cloud
☐ von Cloud in lokales Rechenzentrum
☐ zwischen Clouds

Wie dauerhaft ist die Migration des DBL? (Einfachauswahl)
☐ dauerhaft ☐ temporär

In welcher Zeitspanne findet die Migration statt? (Einfachauswahl)
☐ Stichtagsumstellung (Big-Bang) ☐ längerer Zeitraum (schrittweise)

Wie variabel ist die Ressourcennutzung des DBL im Zielsystem? (Einfachauswahl)
☐ zyklisch variabel ☐ azyklisch variabel ☐ statisch

Wie schnell kann die Ressourcenauslastung des DBL ansteigen? (Einfachauswahl)
☐ schnell ☐ langsam

4.8.2 Cloud Data Hosting Solution Identifikation

Ausgehend von der vorherigen Auswahl der Szenarien, sollen nun Cloud Data Hosting Solutions ermittelt werden um später Konflikte mit den ausgewählten Szenarien zu identifizieren (siehe Abschnitt 5.3.1).

Um welches Cloud Deployment Model für den Zieldatenspeicher soll es sich handeln? (Einfachauswahl)
☐ Private Cloud ☐ Public Cloud
☐ Community Cloud ☐ Hybrid Cloud

An welchem Ort soll das Zielsystem sich befinden? (Einfachauswahl)
☐ On-Premise ☐ Off-Premise

Um welches Cloud Service Model soll es sich beim Zielspeicher handeln? (Einfachauswahl)
☐ SaaS ☐ PaaS ☐ IaaS

Muss der Zieldatenspeicher auf einer verteilten Infrastruktur aufbauen? (Einfachauswahl)
☐ Ja ☐ Nein

Muss der Zieldatenspeicher kompatibel zum Quelldatenspeicher sein? (Einfachauswahl)[7]
☐ Ja ☐ Nein

Neben den oben genannten Fragen zur Identifikation der gewünschten Cloud Data Hosting Solution werden zusätzlich die Kriterien aus Abschnitt 4.8.4 abgefragt um später die Liste an passenden Cloud Data Stores weiter einzuschränken.

4.8.3 Beschreibung des Quellsystems und deren benutzter Funktionen sowie der nutzenden Anwendungen

Um später Konflikte zwischen Quell- und Zielsystem zu identifizieren und benötigte Cloud Data Patterns vorzuschlagen, müssen die Charakteristiken des Quellsystems sowie die Funktionen, welche von darauf zugreifenden Anwendungen genutzt werden, identifiziert werden.

An welchem Ort befindet sich das Quellsystem? (Einfachauswahl)
☐ On-Premise ☐ Off-Premise

[7] Um eine automatische Auswertung vorzunehmen, ob der Quell- und Zieldatenspeicher kompatibel ist wird im Cloud Datenmigrationstool hier das Produkt und die Version des Zieldatenspeichers abgefragt.

Um welches Cloud Deployment Model handelt es sich bei dem Quellsystem? (Einfachauswahl)

- ☐ Keines (nicht in einer Cloud betrieben)
- ☐ Private Cloud ☐ Public Cloud
- ☐ Community Cloud ☐ Hybrid Cloud

Um welches Cloud Service Model handelt es sich bei dem Quellsystem? (Einfachauswahl)

- ☐ Keines (nicht in einer Cloud betrieben)
- ☐ SaaS ☐ PaaS ☐ IaaS

Läuft das Quellsystem auf einer verteilten Infrastruktur? (Einfachauswahl)

- ☐ Ja ☐ Nein

Bietet das Quellsystem eine Synchronisationsfunktion an? (Einfachauswahl)

- ☐ Nein
- ☐ Ja, Einweg-Synchronisation
- ☐ Ja, Zweiweg-Synchronisation

Um welchen Datenspeichertyp handelt es sich bei dem Quellsystem? (Einfachauswahl)

- ☐ RDBMS ☐ NoSQL ☐ BLOB

Bietet das Quellsystem anbieterspezifische Erweiterungen zum SQL 1999 Standard und werden diese von der Anwendung benutzt? (Einfachauswahl)

- ☐ Ja ☐ Nein

Unterstützt das Quellsystem Stored Procedures oder Trigger und werden diese von der Anwendung benutzt? (Einfachauswahl)

- ☐ Ja ☐ Nein

Unterstützt das Quellsystem Transaktionen und werden diese von der Anwendung benutzt? (Einfachauswahl)

- ☐ Ja ☐ Nein

Unterstützt das Quellsystem Joins und werden diese von der Anwendung benutzt? (Einfachauswahl)

- ☐ Ja ☐ Nein

Wird das Quellsystem auf Netzwerkebene über eine per DNS auflösbare URL angesprochen? (Einfachauswahl)

- ☐ Ja ☐ Nein

4.8.4 Identifikation Möglicher Anbieter inkl. benötigter Cloud Data Patterns

Basierend auf der Beschreibung der gewünschten Cloud Data Hosting Solution wird eine Auswahl an möglichen Diensten von bestimmten Cloud Anbietern erstellt (genannt Cloud Data Stores). Da nicht alle Dienste jegliche für die Migrationsszenarien benötigten Funktionen beinhalten, können Cloud Data Patterns eingesetzt werden um fehlende Funktionalität nachzubilden.

Um zuvor Dienste von Cloud Anbietern zu charakterisieren, dient der folgende Fragebogen. Die Fragen dazu wurden aus den technischen Beschreibungen verschiedener Cloud Data Stores abgeleitet und erweitert um Fragen die in Diskussionen mit dem Betreuer dieser Arbeit und einem anderen Studenten aufkamen, der an einem verwandten Thema arbeitet.

Schritt Eins: Cloud Dienst Charakterisieren

Name des Dienstes: _____
Anbieter des Dienstes: _____
Webseite des Dienstes: _____
Beschreibung des Dienstes: _____

Skalierbarkeit
Skaliert der Dienst automatisch? (Einfachauswahl)
☐ Ja ☐ Nein ☐ Ja, zum Teil

Wie skaliert der Dienst?
☐ Vertikal (scale up)
☐ Horizontal (scale out)
☐ Vertikal (scale up) und Horizontal (scale out)

Ist der Dienst skalierbar? (Einfachauswahl)
☐ Voll skalierbar
☐ Begrenzt skalierbar, Grenzen: _____
☐ Nicht skalierbar

Wie lange dauert es, bis eine neue Instanz verfügbar ist?
☐ Nicht möglich ☐ Minuten: _____

Nach welchem Kriterium wird automatisch skaliert?
☐ Keine Skalierung ☐ System Load ☐ Latenz

Verfügbarkeit
Arbeitet der Dienst auf einer verteilten Infrastruktur?
☐ Ja ☐ Nein

Welche Art von Replikation wird dabei verwendet? (Einfachauswahl)
☐ Master/Master Replikation
☐ Master/Slave Replikation

Wie geschieht die Replikation?
☐ Synchron ☐ Asynchron

An welchem Ort findet die Replikation statt? (Einfachauswahl)
☐ Gleiches Rechenzentrum („same Availability Zone“)
☐ Anderes Rechenzentrum im gleichen Ort („same Region“)
☐ Anderes Rechenzentrum an anderem Ort
☐ Keine Replikation

Bietet der Dienst automatisches Failover an? (Einfachauswahl)
☐ Ja ☐ Nein

Welche Verfügbarkeit garantiert der Dienstanbieter? (Einfachauswahl)
☐ < 99,0 % ☐ zw. 99,0 und 99,9 % ☐ > 99,9 %

Sicherheit
Speichert der Dienst die Daten verschlüsselt?
☐ Ja ☐ Nein

Erlaubt der Dienst einen Transfer der Daten in verschlüsselter Form?
☐ Ja ☐ Nein

Erlaubt der Dienst die Konfiguration der Firewall?
☐ Ja ☐ Nein

Erlaubt der Dienst Authentifizierung?
☐ Ja ☐ Nein

Erlaubt der Dienst Autorisierung?
☐ Ja ☐ Nein

Ist der Dienst geeignet um vertrauliche/persönliche Daten zu speichern? (Einfachauswahl)
☐ Ja ☐ Nein

Garantiert der Dienst die Integrität der Daten?
☐ Ja ☐ Nein

Ort
Gibt es eine Wahl des Speicherortes der Daten?
☐ Ja
☐ Nein (es steht nur ein Ort zur Verfügung)
☐ Nein (automatisch gewählt)

An welchem Ort werden die Daten gespeichert?
☐ Region/Land/Kontinent: _____

Wo befindet sich der Cloud Dienst?
☐ On-Premise ☐ Off-Premise

Datenbeschränkungen
Gibt es Einschränkungen bezüglich der max. Objekt- (NoSQL) / Zeilen- (RDBMS) / Dateigröße (BLOB)
☐ Ja, Byte: _____

Gibt es Einschränkungen bezüglich der max. Domain (NoSQL) / Tabellen (RDBMS) / Bucketgröße (BLOB)

☐ Ja, GB: _____

Gibt es Einschränkungen bezüglich der max. Objekt/Zeilen/Dateien Anzahl pro Instanz?

☐ Ja, Zahl: _____

Gibt es Einschränkungen bezüglich der max. Instanzgröße (alle Domains, alle Tabellen, alle Buckets)

☐ Ja, GB: _____

Ist die Instanzgröße vorgegeben?

☐ Ja ☐ Nein

Interoperabilität

Bietet der Dienst eine Synchronisationsfunktion an? (Mehrfachauswahl)

☐ Nein
☐ Import ☐ Export
☐ Einweg-Synchronisation ☐ Zweiweg-Synchronisation
☐ Keine Synchronisation

Welches Austauschformat bietet der Dienst an?

☐ XML ☐ JSON ☐ Proprietär

Welche Schnittstellen bietet der Dienst an?

☐ SOA ☐ REST-API ☐ SQL ☐ Proprietär

Welches ORM Verfahren wird unterstützt?

☐ JPA ☐ JDO ☐ LINQ

Bietet der Anbieter Unterstützung bei der Migration und dem Deployment?

☐ Ja ☐ Nein

Unterstützt der Dienst ein bestimmtes Integrated Development Environment (IDE)?

☐ Eclipse ☐ NetBeans
☐ Visual Studio ☐ IntelliJ IDEA

Gibt es für den Dienst Developer SDKs?

☐ Java ☐ .Net ☐ PHP
☐ Ruby ☐ Python
☐ Keine ☐ Keine benötigt (da Standard Support)

Kompatibilität

Welches Standardprodukt inkl. Version verwendet der Dienst?

☐ MySQL
☐ PostgreSQL
☐ MS SQL (SQL Server)
☐ DB2
☐ Version: _____

Speicher
Ist der Zugriff einschränkbar?
☐ Ja, VPN

Um welchen Datenspeichertyp handelt es sich bei dem Dienst? (Einfachauswahl)
☐ RDBMS ☐ NoSQL ☐ BLOB ☐ CDN

Unterstützt der Speicher Transaktionen?
☐ Ja, ACID-konform

Performanz
Ist die Antwortzeit des Dienstes (Read/Write/Response) konstant und vorhersagbar?
☐ Nicht konstant ☐ Vorhersagbar, Millisekunden: _____

Welche Transferbeschränkung hat der Dienst?
☐ GB In/Out: _____

Welche Daten Durchsatzbeschränkung hat der Dienst?
☐ Bytes/Sekunde: _____

Welche Latenz hat der Dienst?
☐ Millisekunden: _____

CAP (consistency, availability, partition tolerance)
Welche Konsistenz beim Lesen der Daten bietet der Dienst? (Einfachauswahl)
☐ Weak Consistency
☐ Strict Consistency
☐ Eventual Consistency

Ist der Dienst auch bei Netzwerkausfällen verfügbar? (Einfachauswahl)
☐ Ja ☐ Nein

Flexibilität
Ist ein Schema festlegbar?
☐ Ja ☐ Nein

Ist das Schema anpassbar?
☐ Ja ☐ Nein

Cloud Computing
Um welches Cloud Service Model handelt es sich bei dem Dienst? (Einfachauswahl)
☐ SaaS ☐ PaaS ☐ IaaS

Um welches Cloud Deployment Model handelt es sich bei dem Dienst? (Einfachauswahl)
☐ Private Cloud ☐ Public Cloud
☐ Community Cloud ☐ Hybrid Cloud

Management- und Wartungsaufwand
In wieweit ist der Dienst automatisiert?
☐ Self-Service ☐ Gemanagt/komplett automatisiert

Monitoring
Welche KPIs werden unterstützt?

- ☐ Processing Load (CPU)
- ☐ Data Transfer - Network (I/O)
- ☐ Data Transfer - Disk (I/O)
- ☐ Memory Load (RAM)

Backup
In welchem Intervall können Backups erstellt werden?

- ☐ Keine Backups möglich
- ☐ Periodisch, alle x Minuten: _____
- ☐ Auf Anfrage

Ist während des Backups der Dienst kurzzeitig nicht erreichbar?

- ☐ Ja ☐ Nein

Wie viele Backups werden vom Dienst aufbewahrt?

- ☐ None
- ☐ Anzahl Backups: _____
- ☐ Backups für x Tage: _____

Welche Backup Methode wird verwendet?

- ☐ Schnappschuss (Dateisystem)
- ☐ Datei-Backup (Export)
- ☐ Inkrementelles Backup (Log Dateien)

Mandantenfähigkeit
Bietet der Dienst eine Mandantenfunktion an?

- ☐ Ja ☐ Nein

Welchen Typ von Mandantenfähigkeit unterstützt der Dienst?

- ☐ Multiple Instanzen ☐ Native Mandantenfähigkeit

Schritte Zwei: Ermittlung der Konflikte

Sofern ein Zielsystem nicht alle Funktionen des Quellsystems und der zusätzlich benötigten Funktionen bzgl. des gewählten Szenarios unterstützt, müssen Cloud Data Patterns und weitere Anpassungen vorgeschlagen werden, die diese Konflikte beheben können.

Kriterium	**Konflikt**		**Cloud Data Pattern / anderer Lösungsvorschlag**
	Quellsystem/ Szenarioanforderung	**Zielsystem**	
Stored Procedure / Trigger	Ja	Nein	Emulator of Stored Procedures, Anpassung des DAL, Anpassung der Anwendungslogik
Kompatibilität bzw. Versions- / Produktunterschied	gleiches Produkt, unterschiedliche Version		Data Store Functionality Extension, Anpassung des DAL, Anpassung der Anwendungslogik
	unterschiedliches Produkt		
Skalierbarkeit[8]	Ja	Nein	Local Database Proxy, Anpassung des DAL, Anpassung der Anwendungslogik
Sharding[9]	Ja	Nein	Local Sharding-Based Router, Anpassung des DAL, Anpassung der Anwendungslogik
Vertraulichkeit des Speichers	Ja	Nein	Confidentiality Level Data Aggregator[10], Filter of Critical Data, Pseudonymizer of Critical Data, Anonymizer of Critical Data, Anpassung des DAL, Anpassung der Anwendungslogik
Schneller Anstieg des Ressourcenbedarfs	Ja	Nein	Pool an Reserveinstanzen
Synchronisation (Einweg/Zwei-Wege)	Ja	Nein	ETL Tool oder andere externe Datenmigrationstools, Anpassung des DAL, Anpassung der Anwendungslogik
Verteilte Infrastruktur (Hochverfügbarkeit, Hochskalierbarkeit)	Ja	Nein	Instanziierung mehrere Instanzen des Zielsystems und Einrichtung einer Synchronisation, Backups

[8] bezogen auf Lesezugriffe

[9] bezogen auf Skalierbarkeit der Lese- und Schreibzugriffe

[10] nur im Szenario „Datenimport aus der Cloud“ relevant, da nur hier mehrere Quellsysteme existieren

			und Failovers
Ort des Quell- und Zielsystems	inkompatibel		Wahl eines anderen Zielsystems
Cloud Deployment Model	inkompatibel		Wahl eines anderen Zielsystems
Cloud Service Model	inkompatibel		Wahl eines anderen Zielsystems
Änderung der IP-Adresse	per URL adressiert (DNS)	kompatibel, erreichbar unter neuer IP Adresse	Änderung des CNAME Eintrags und Anpassung der DNS Cache TTL, Konfiguration der Firewall, Änderung der Verbindungskennung
	per IP adressiert	kompatibel, erreichbar unter neuer IP Adresse	Änderung der Verbindungskennung, Konfiguration der Firewall
Komplexe Anfragen	RDBMS	NoSQL	Nutzung externer Dienste für komplexe Abfragen, Anpassung des DAL, Anpassung der Anwendungslogik

Tabelle 6 Übersicht von Konflikten und deren Lösungen durch Cloud Data Patterns

Die in Tabelle 6 gezeigten Konflikte leiten sich aus den Antworten der vorherigen Fragebögen (Identifikation des Szenarios, Beschreibung des Quellsystems, Beschreibung der möglichen Zielsysteme) direkt ab. Zu jedem Konflikt werden außerdem ein oder mehrere Lösungsvorschläge gemacht.

Sofern die Konfliktlösung „Wahl eines anderen Zielsystems“ lautet, müssen weitere passendere Zielsysteme identifiziert und beschrieben werden. Alternativ können auch sekundäre Anforderungen bei der Szenario Beschreibung entfernt werden. Dies kann z.B. durch eine Priorisierung der Kriterien erfolgen.

5 Cloud Datenmigrations-Tool

Das Cloud Datenmigrations-Tool soll bei der Migration der Datenschicht in die Cloud unterstützen. Dabei soll sowohl die Vorbereitung der Migration, als auch die Migration an sich prototypisch bewerkstelligt werden.

5.1 Analyse

Die Analyse erfasst in Kurzform alle funktionalen und nicht-funktionalen Anforderungen an das Cloud Data Migration Tool und listet diese in Tabelle 7 und Tabelle 8 auf. Die Anforderungen stammen vom Betreuer dieser Arbeit.

5.1.1 Funktionale Anforderungen

Nr.	Anforderung	Beschreibung
FR-1	Englischsprachige Benutzer-oberfläche	-
FR-2	Entscheidungsunterstützung bei der Datenmigration in die Cloud	Wahl eines Scenarios mit Ausprägungen (inkl. Konflikterkennung), Beschreibung der gewünschten Data Hosting Solution und des Quellsystems, Vorschlag vorhandener Cloud Dienste mit ggf. notwendigen Cloud Data Pattern und Auswirkungen auf die Netzwerk-, Data Access- und Anwendungsebene (siehe Abschnitt 4.8).
FR-3	Unterstützung der Cloud Data Hosting Solutions Amazon EC2 (DBMS on Image), Google Cloud SQL, Azure SQL drei unterschiedlichen während der Migration	Die Anwendung soll die Migration (siehe Abschnitt 4.7) des DBL weitgehend automatisch übernehmen. Alternativ steht auch noch Amazon RDS oder der Google App Engine Datastore zur Verfügung.
FR-4	Unterstützung des Szenarios „Reines Outsourcing“	Bei der Migration (siehe Abschnitt 4.7) soll das Szenario „Reines Outsourcing“ unterstützt werden. Zusätzlich ist auch das „Cloud Burst“ und „Hochskalierbarer Datenspeicher“ Szenario wünschenswert aber nicht zwingend notwendig.
FR-5	Katalogverwaltung von Cloud Data Hosting Solutions	Es sollen verschiedene Dienste von Cloud Anbietern als mögliche Cloud Data Hosting Solution beschrieben (siehe Abschnitt 4.8) und angezeigt werden können.
FR-6	Report des Migrationsprojektes	Die getroffenen Auswahlen, resultierenden Konflikte und Lösungen sollen als Report druckbar sein.
FR-7	Projektverwaltung	Jeder am System angemeldete Nutzer soll mehrere Migrationsprojekte anlegen und verwalten können.

Tabelle 7 Funktionale Anforderungen an das Cloud Data Migration Tool

5.1.2 Nicht-Funktionale Anforderungen

Nr.	Anforderung	Beschreibung
NFR-1	Benutzerfreundlich und Intuitiv	Die Anwendung soll nach den Maßstäben der Gebrauchstauglichkeit entwickelt werden (Anforderungsangemessenheit, Lernförderlichkeit, Individualisierbarkeit, Fehlertoleranz, Erwartungskonformität, Steuerbarkeit, Selbstbeschreibungsfähigkeit)
NFR-2	Erweiterbarkeit	Gemäß der erweiterbaren Methodik soll auch die Anwendung erweiterbar sein, um z.B. die Cloud Data Hosting Solutions in einer Policy Sprache (WS-Policy) zu beschreiben oder um Security oder Compliance Kriterien zu berücksichtigen.
NFR-3	Programmiersprache Java	Um die Integration mit anderen angrenzenden Entwicklungen zu erleichtern soll die Anwendung in der gleichen Sprache (Java) wie die angrenzenden Entwicklungen geschrieben werden.
NFR-4	Open Source Lizenz	Um eine freie Erweiterbarkeit der Anwendung durch beliebige weitere Personen zu gewährleisten und kompatibel zu anderen verwendeten Lizenzen innerhalb des Instituts zu sein, muss die Anwendung unter der Apache License Version 2.0 veröffentlicht werden.
NFR-5	Installationsanleitung	Schritt-für-Schritt Anleitung oder Screencast der zeigt wie die Anwendung installiert wird.

Tabelle 8 Nicht-Funktionale Anforderungen an das Cloud Data Migration Tool

5.2 Spezifikation

Aus den Anforderungen der Analyse werden im Folgenden Use Cases abgeleitet und näher spezifiziert.

5.2.1 Use Case Diagramm

Das folgende Use Case Diagramm (Abbildung 30) listet zwei grundsätzlich verschiedene Akteure: Softwareentwickler und Data Store Experte. Der Softwareentwickler möchte den DBL einer bestehenden Anwendung in die Cloud migrieren und benötigt dazu Hilfe. Der Data Store Experte kennt sich mit den verschiedenen Datendiensten der Cloud Anbieter aus und kann diese genau beschreiben.

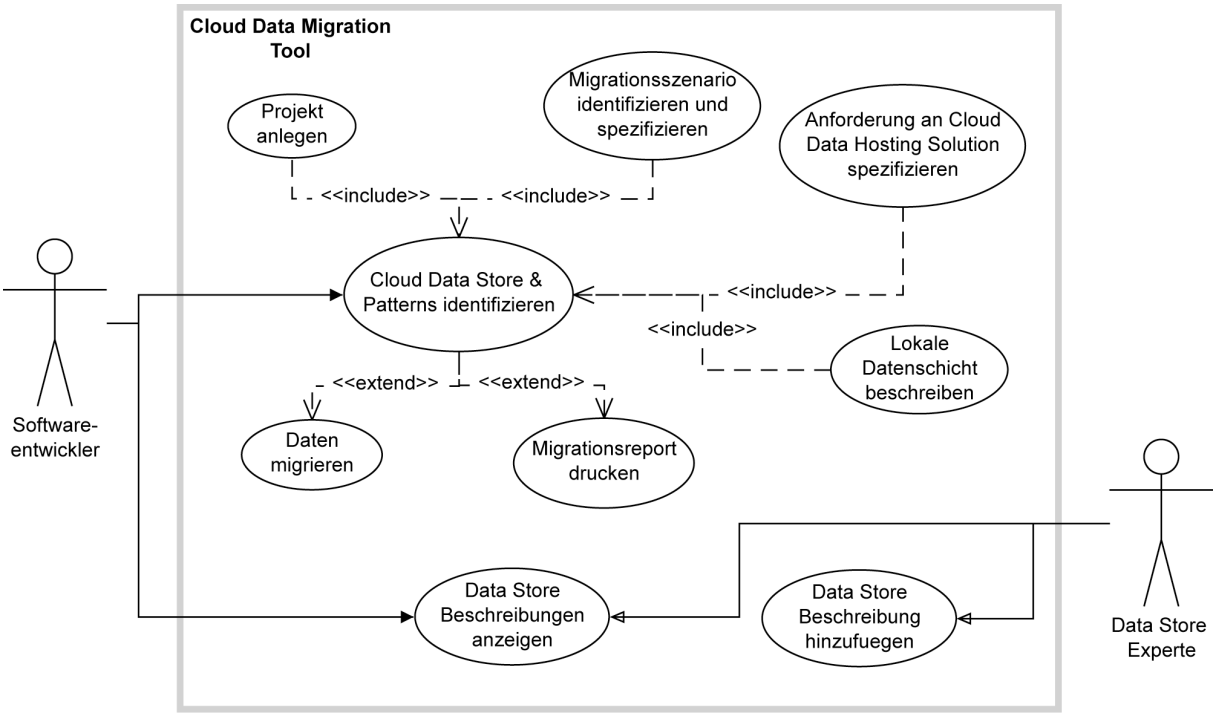

Abbildung 30 Use Case Diagramm des Cloud Data Migration Tools

5.2.2 Use Case 1: Data Store Beschreibungen Anzeigen (FR-5)

Akteur	Softwareentwickler, Data Store Experte
Ziel	Der Softwareentwickler oder Data Store Experte möchte eine Übersicht aller verfügbaren Data Stores bekommen und sich die Details ansehen.
Vorbedingung	Es ist mindestens ein Data Store gespeichert.
Nachbedingung	Der Softwareentwickler oder Data Store Experte hat eine Übersicht aller verfügbaren Data Stores bekommen und sich einige davon im Detail angesehen.
Nachbedingung im Sonderfall	-
Regulärer Ablauf	Der Softwareentwickler oder Data Store Experte wechselt zur "Data Store" Übersicht. Das System zeigt dem Softwareentwickler oder Data Store Experte eine Liste aller gespeicherten Data Stores an. Der Softwareentwickler oder Data Store Experte wählt einen Data Store aus. Das System zeigt alle Details zum ausgewählten Data Store an.
Alternativer Ablauf	-
Sonderfall	-

Tabelle 9 Use Case 1: Data Store Beschreibungen anzeigen

5.2.3 Use Case 2: Data Store Beschreibung Hinzufügen (FR-5)

Akteur	Data Store Experte
Ziel	Der Data Store Experte möchte dem Softwareentwickler neue Data Stores als Zielsystem für seine Datenmigration zur Verfügung stellen.
Vorbedingung	Alle funktionalen und nicht-funktionalen Kriterien laut Fragebo-

	gen (Abschnitt 4.8.4, Schritt 1) sind ermittelt. Der Data Store Experte ist im System angemeldet.
Nachbedingung	Ein neuer Data Store wurde hinzugefügt und kann als mögliches Zielsystem ausgewählt werden.
Nachbedingung im Sonderfall	5a/5b Der Data Store Experte ist über das Problem informiert und kann das Formular nach der Behebung erneut absenden. 6a Der Data Store Experte ist über das Problem informiert und kann erneut versuchen das Formular abzusenden oder seine Tätigkeit abbrechen.
Regulärer Ablauf	1. Der Data Store Experte wechselt zum "Data Store Hinzufügen" Formular. 2. Das System zeigt dem Data Store Experte ein leeres Formular mit allen Fragen lauf Fragebogen (siehe Abschnitt 4.8.4) an. 3. Der Data Store Experte füllt alle Felder des Formulars aus. 4. Der Data Store Experte sendet das Formular an das System. 5. Das System prüft das Formular auf Vollständigkeit und Konfliktfreiheit. 6. Das System speichert den neuen Data Store. 7. Das System zeigt dem Data Store Experten einen Hinweis, dass sein neuer Data Store gespeichert wurde. 8. Das System zeigt dem Data Store Experten eine Liste aller im System befindlichen Data Stores.
Alternativer Ablauf	-
Sonderfall	5a. Das System findet eine nicht beantwortete Frage und zeigt dem Data Store Experten das vorausgefüllte Formular mit einem entsprechenden Hinweis an. 5b. Das System entdeckt einen Konflikt und zeigt dem Data Store Experten das vorausgefüllte Formular mit einem entsprechenden Hinweis an. 6a. Das System kann den Data Store nicht speichern (keine DB-Verbindung) und zeigt dem Data Store Experten das vorausgefüllte Formular mit einem entsprechenden Hinweis an.

Tabelle 10 Use Case 2: Data Store Beschreibung hinzufügen

5.2.4 Use Case 3: Projekt anlegen (FR-7)

Akteur	Softwareentwickler
Ziel	Der Softwareentwickler möchte seine Arbeiten in einem Projekt speichern.
Vorbedingung	Der Softwareentwickler ist im System angemeldet.
Nachbedingung	Der Softwareentwickler hat ein Projekt angelegt.
Nachbedingung im Sonderfall	Der Softwareentwickler hat ein Projekt angelegt.
Regulärer Ablauf	1. Der Softwareentwickler wechselt zum "Projekt Hinzufügen" Formular. 2. Das System zeigt dem Softwareentwickler ein leeres Formular an. 3. Der Softwareentwickler füllt alle Felder des Formulars aus.

	4. Der Softwareentwickler sendet das Formular an das System. 5. Das System prüft das Formular auf Vollständigkeit. 6. Das System speichert das neue Projekt. 7. Das System zeigt dem Softwareentwickler einen Hinweis, dass sein neues Projekt gespeichert wurde. 8. Das System zeigt dem Softwareentwickler eine Liste aller im System befindlichen Projekte.
Alternativer Ablauf	-
Sonderfall	5a. Bei Unvollständigkeit des Formulars zeigt das System dem Softwareentwickler das vorausgefüllte Formular mit einem entsprechenden Hinweis an. Der Softwareentwickler fährt dann bei Schritt 3 fort und korrigiert die Angaben um das Formular erneut abzusenden, bis das System das Projekt speichern konnte.

Tabelle 11 Use Case 3: Projekt anlegen

5.2.5 Use Case 4: Migrationsszenario Identifizieren und Spezifizieren (FR-2, FR-4)

Der folgende Use Case ermöglicht es ein einzelnes Migrationsszenario pro Projekt auszuwählen. Spielen mehrere Migrationsszenarien innerhalb eines Migrationsprojekts eine wesentliche Rolle, so muss für jedes Szenario ein einzelnes Projekt erstellt werden.

Akteur	Softwareentwickler
Ziel	Der Softwareentwickler möchte sich allen Ausprägungen seiner Daten Migration bewusst werden und potentielle Konflikte lösen.
Vorbedingung	Der Softwareentwickler ist im System angemeldet.
Nachbedingung	Der Softwareentwickler hat ein Migrationsszenario ausgewählt und dessen Ausprägungen spezifiziert. Es bestehen keine Konflikte zwischen dem ausgewählten Migrationsszenario und den vorgestellten Ausprägungen einer Migration.
Nachbedingung im Sonderfall	-
Regulärer Ablauf	1. Der Softwareentwickler wechselt zum "Migrationsszenario auswählen" Formular. 2. Das System zeigt dem Softwareentwickler ein leeres Formular mit allen Migrationsszenarien (siehe Abschnitt 4.8.1) an. 3. Der Softwareentwickler wählt ein relevantes Migrationsszenario im Formular aus. 4. Der Softwareentwickler sendet das Formular an das System. 5. Das System prüft das Formular auf Vollständigkeit. 6. Das System speichert das ausgewählte Migrationsszenario. 7. Das System zeigt dem Softwareentwickler einen Hinweis, dass seine Auswahl gespeichert wurde. 8. Das System zeigt dem Softwareentwickler ein Formular mit Kriterien und möglichen Ausprägungen von Datenmigrationen. 9. Der Softwareentwickler wählt seine gewünschten Ausprägungen aus.

	10. Der Softwareentwickler sendet das Formular an das System. 11. Das System prüft das Formular auf Vollständigkeit und ermittelt Konflikte. 12. Das System speichert die ausgewählten Kriterien. 13. Das System zeigt dem Softwareentwickler einen Hinweis, dass seine Auswahl gespeichert wurde. 14. Das System zeigt dem Softwareentwickler die Projektübersicht mit potentiellen Konflikten bezogen auf die Migrationsszenario-Auswahl (Schritt 3) und deren Ausprägungen (Schritt 9) an. Weiter listet das System mögliche Anpassungsmöglichkeiten auf Netzwerk-, Data Access- und Anwendungsebene auf Grund der Auswahl des Migrationsszenarios (Schritt 3) auf. 15. Der Softwareentwickler kann die Konflikte gemäß den vorgeschlagenen Lösungen beheben, indem er Schritt 3 oder 9 wiederholt.
Alternativer Ablauf	-
Sonderfall	-

Tabelle 12 Use Case 4: Migrationsszenario identifizieren und spezifizieren

5.2.6 Use Case 5: Anforderung an Cloud Data Hosting Solution Spezifizieren (FR-2)

Der folgende Use Case ermöglicht es eine einzelne Cloud Data Hosting Solution pro Projekt auszuwählen. Spielen mehrere Cloud Data Hosting Solutions innerhalb eines Migrationsprojekts eine wesentliche Rolle, so muss für jede Cloud Data Hosting Solution ein einzelnes Projekt erstellt werden.

Akteur	Softwareentwickler
Ziel	Der Softwareentwickler möchte sich allen Ausprägungen seiner Cloud Data Hosting Solution bewusst werden und alle Anforderungen spezifizieren. Aufbauend auf diesen Anforderungen sollen später passende Cloud Data Stores identifiziert werden können.
Vorbedingung	Der Softwareentwickler ist im System angemeldet.
Nachbedingung	Der Softwareentwickler hat seine Anforderungen an die Cloud Data Hosting Solution spezifiziert um eine Auswahl an passenden Diensten und notwendigen Cloud Data Patterns zu erhalten.
Nachbedingung im Sonderfall	-
Regulärer Ablauf	1. Der Softwareentwickler wechselt zur Projektübersicht. 2. Das System zeigt einen Überblick über das Datenmigrationsprojekt mit verschiedenen zuvor getroffenen Entscheidungen. 3. Der Softwareentwickler teilt dem System mit, dass er eine Cloud Data Hosting Solution auswählen möchte. 4. Das System zeigt dem Softwareentwickler ein leeres Formular mit allen Fragen lauf Fragebogen (siehe Abschnitt 4.8.2 und 4.8.4) an. 5. Der Softwareentwickler wählt alle relevanten Anforderungen an die Cloud Data Hosting Solution im Formular aus.

	6. Der Softwareentwickler sendet das Formular an das System. 7. Das System prüft das Formular auf Vollständigkeit. 8. Das System speichert die ausgewählten Anforderungen an die Cloud Data Hosting Solution. 9. Das System zeigt dem Softwareentwickler einen Hinweis, dass seine Auswahl gespeichert wurde. 10. Das System zeigt dem Softwareentwickler die Projektübersicht an.
Alternativer Ablauf	-
Sonderfall	-

Tabelle 13 Use Case 5: Anforderung an Cloud Data Hosting Solution spezifizieren

5.2.7 Use Case 6: Cloud Data Store und Patterns Identifizieren (FR-2)

Akteur	Softwareentwickler
Ziel	Der Softwareentwickler möchte passende Dienste aufgelistet bekommen, die als Zielsystem für sein Cloud Datenmigration in Frage kommen. Bei Konflikten zwischen seinen Anforderungen und den Spezifikationen der Dienste sollen Lösungen u.a. in Form von Cloud Data Patterns angeboten werden.
Vorbedingung	Der Softwareentwickler ist im System angemeldet, hat ein Projekt angelegt, sein Szenario spezifiziert und Anforderungen an das Zielsystem festgelegt.
Nachbedingung	Der Softwareentwickler hat eine Liste möglicher Dienste für seine Migration inkl. benötigter Anpassungen in Form von Cloud Data Patterns oder Ähnlichem.
Nachbedingung im Sonderfall	Der Softwareentwickler muss seine Anforderungen anpassen bzw. entschärfen um Konflikte zu lösen.
Regulärer Ablauf	1. Der Softwareentwickler wechselt zur Projektübersicht. 2. Das System zeigt einen Überblick über das Datenmigrationsprojekt mit verschiedenen zuvor getroffenen Entscheidungen. 3. Der Softwareentwickler fordert das System auf einen Cloud Data Store vorzuschlagen. 4. Das System ermittelt kompatible und teilweise inkompatible Cloud Data Stores. Zu den teilweise inkompatiblen Cloud Data Stores werden die Inkompatibilitäten hervorgehoben. 5. Der Softwareentwickler wählt einen Cloud Data Store aus. 6. Der Softwareentwickler fordert das System auf die Angaben zu speichern. 7. Das System speichert die Auswahl des Cloud Data Stores. 8. Das System zeigt die Projektübersicht mit potentiellen Konflikten und Pattern als Lösungsmöglichkeiten an, welche aus der Auswahl des Cloud Data Stores resultieren.
Alternativer Ablauf	-
Sonderfall	-

Tabelle 14 Use Case 6: Cloud Data Store und Patterns identifizieren

5.2.8 Use Case 7: Daten Migrieren (FR-3)

Für den Fall, dass mehrere Datenmigrationen stattfinden, muss der Use Case 7 mehrfach hintereinander ausgeführt werden.

Akteur	Softwareentwickler
Ziel	Der Softwareentwickler möchte möglichst automatisiert die Daten aus seinem Quellsystem in das Zielsystem migrieren.
Vorbedingung	Der Softwareentwickler ist im System angemeldet, hat ein Projekt angelegt, sein Quellsystem beschrieben und einen Cloud Data Store ausgewählt. Der Softwareentwickler hat die Zugangsdaten zum Quellsystem und hat sich beim Cloud Anbieter registriert und das Zielsystem eingerichtet und konfiguriert.
Nachbedingung	Der Softwareentwickler hat den DBL des Quellsystems in das Zielsystem migriert und die darüber liegenden Schichten so angepasst, dass sie auf den migrierten DBL zugreifen.
Nachbedingung im Sonderfall	-
Regulärer Ablauf	1. Der Softwareentwickler wechselt zur Projektübersicht. 2. Das System zeigt einen Überblick über das Datenmigrationsprojekt mit verschiedenen zuvor getroffenen Entscheidungen. 3. Das System zeigt dem Softwareentwickler ein Formular zur Eingabe der Zugangsdaten zum Quell- und Zielsystem. 4. Der Softwareentwickler gibt die Zugangsdaten des Quell- und Zielsystems an. 5. Der Softwareentwickler fordert das System auf die Daten zu migrieren. 6. Das System prüft die Zugangsdaten des Quell- und Zielsystems und versucht jeweils eine Verbindung herzustellen. 7. Das System kopiert die Daten vom Quell- zum Zielsystem. 8. Das System teilt dem Softwareentwickler den Fortschritt der Migration mit und benachrichtigt den Softwareentwickler, dass die Migration abgeschlossen wurde. 9. Das System gibt dem Softwareentwickler weitere Anweisungen, wie er die Anwendung, dessen DBL migriert wurde, anpassen muss, um den neuen Zielspeicher zu verwenden.
Alternativer Ablauf	-
Sonderfall	-

Tabelle 15 Use Case 7: Daten migrieren

5.2.9 Use Case 8: Migrationsreport Drucken (FR-6)

Akteur	Softwareentwickler
Ziel	Der Softwareentwickler möchte alle seine getroffenen Angaben inkl. daraus resultierender Konflikte und Lösungsmöglichkeiten in einem Report ausdrucken.
Vorbedingung	Der Softwareentwickler ist im System angemeldet, hat ein Projekt angelegt.

Nachbedingung	Der Softwareentwickler hat eine Auflistung aller getroffenen Angaben und daraus resultierenden Konflikten und Lösungsmöglichkeiten ausgedruckt.
Nachbedingung im Sonderfall	-
Regulärer Ablauf	1. Der Softwareentwickler wechselt zur Projektübersicht. 2. Das System zeigt einen Überblick über das Datenmigrationsprojekt mit verschiedenen zuvor getroffenen Entscheidungen. 3. Der Softwareentwickler fordert das System auf einen Report zu drucken. 4. Das System zeigt die Druckansicht des Reports. 5. Der Softwareentwickler sendet den Druckauftrag ab. 6. Das System druckt den Report.
Alternativer Ablauf	-
Sonderfall	-

Tabelle 16 Use Case 6: Migrationsreport drucken

5.2.10 Use Case 9: Lokale Datenschicht Beschreiben (FR-2)

Akteur	Softwareentwickler
Ziel	Der Softwareentwickler möchte seine lokale Datenschicht beschreiben um weitere Konflikte und deren Lösungsmöglichkeiten vorgeschlagen zu bekommen.
Vorbedingung	Der Softwareentwickler ist im System angemeldet, hat ein Projekt angelegt.
Nachbedingung	Der Softwareentwickler hat seine lokale Datenschicht beschrieben und potentielle Konflikte und deren Lösungsmöglichkeiten erfahren.
Nachbedingung im Sonderfall	-
Regulärer Ablauf	1. Der Softwareentwickler wechselt zur Projektübersicht. 2. Das System zeigt einen Überblick über das Datenmigrationsprojekt mit verschiedenen zuvor getroffenen Entscheidungen. 3. Der Softwareentwickler teilt dem System mit, dass er seine lokale Datenschicht beschreiben möchte. 4. Das System zeigt ein Formular zur Beschreibung der lokalen Datenschicht an. 5. Der Softwareentwickler füllt das Formular aus und fordert das System auf es zu speichern. 6. Das System speichert die Angaben. 7. Das System ermittelt die Konflikte aus der Beschreibung der lokalen Datenschicht und den zuvor gemachten Angaben und zeigt diese in der Projektübersicht an.
Alternativer Ablauf	-
Sonderfall	-

Tabelle 17 Use Case 6: Lokale Datenschicht beschreiben

5.3 Design

Aus den funktionalen und nicht-funktionalen Anforderungen werden im Folgenden Entscheidungen für das Software-Design getroffen.

5.3.1 Architektur

Programmiersprache und Laufzeitumgebung

Das Cloud Data Migration Tool baut auf einer Java 6 Webanwendung (Servlet-Container) auf. Als Container soll die Anwendung sowohl auf Tomcat 7, als auch in der Google App Engine lauffähig sein.

Datenhaltung

Die Datenhaltung soll sowohl in einer MySQL 5.5 Datenbank (InnoDB), als auch später in Google Cloud SQL (basiert auf MySQL) möglich sein. Als ORM soll JDO 3.1 verwendet werden, da dieser ORM sowohl für relationale, als auch NoSQL Datenspeicher konzipiert wurde und das Cloud Data Migration Tool auf verschiedenen Datenspeicher(typen) lauffähig sein soll. DataNucleus 3.0 soll als JDO Implementierung verwendet werden, da diese Implementierung viele relationale Datenspeicher wie MySQL, Oracle und SQL Server, als auch NoSQL Datenspeicher wie Google App Engine Data Store unterstützt.

Komposition

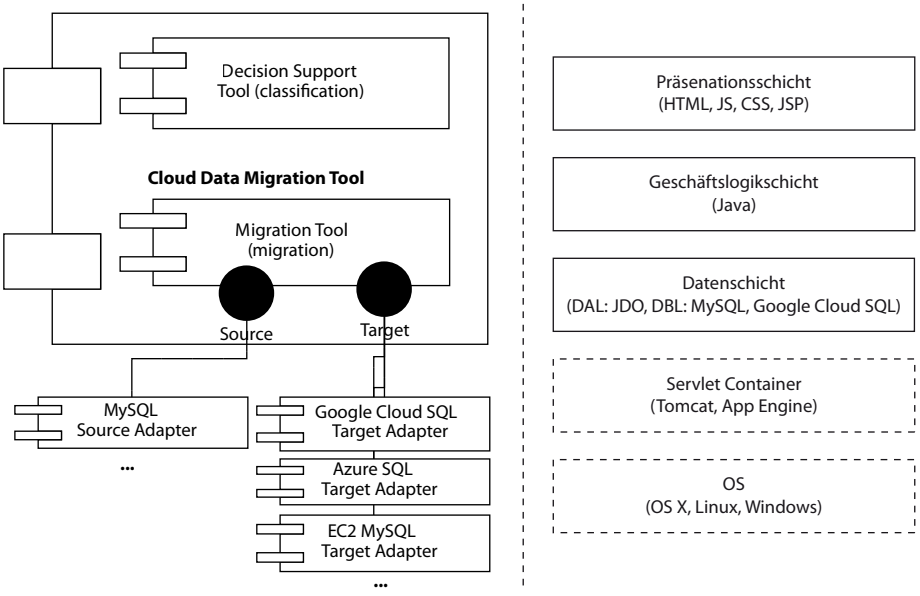

Abbildung 31 UML Komponenten- und Schichtendiagramm des Cloud Data Migration Tools

Das Cloud Data Migration Tool ist, wie in Abbildung 31 dargestellt, modular und erweiterbar aufgebaut. Es besteht aus den zwei Komponenten Decision Support Tool (classification) und Migration Tool (migration). Das Decision Support Tool beinhaltet dabei auch die Erfassung von verschiedenen Cloud Data Stores. Der Aufbau der Anwendung entspricht einer klassischen 3-Schichten Architektur.

Das Migration Tool unterstützt zudem verschiedene Adapter für Quell- und Zielsysteme. Die Adapter können die Datenmigration selbst übernehmen oder auf Tools und Anleitungen verweisen, welche die Migration übernehmen.

Interne Schnittstellen

Das Migration Tool bietet zwei interne Schnittstellen an mit denen Daten aus einem Quellsystem geladen und in ein Zielsystem migriert werden können.

Externe Schnittstellen

Die Adapter des Migration Tools benutzen die Schnittstellen der entsprechenden Quell- und Zielsysteme um Daten aus den Quellsystemen in die Zielsysteme zu migrieren.

Struktur

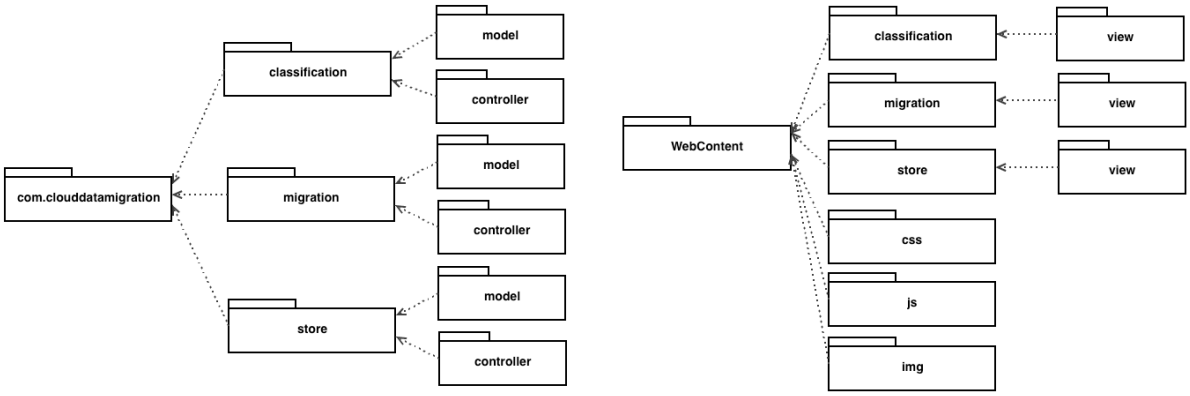

Abbildung 32 UML Paketdiagramm des Cloud Data Migration Tools

Das in Abbildung 32 gezeigte Paketdiagramm zeigt den internen Aufbau des Cloud Data Migration Tools. Die Struktur lehnt sich am Model-View-Controller Pattern an. Dabei werden Views im JSP Format erstellt, Controller werden durch Java Servlets repräsentiert und als Model dienen simple Java Klassen (POJO).

5.3.2 Entitäten und Klassen

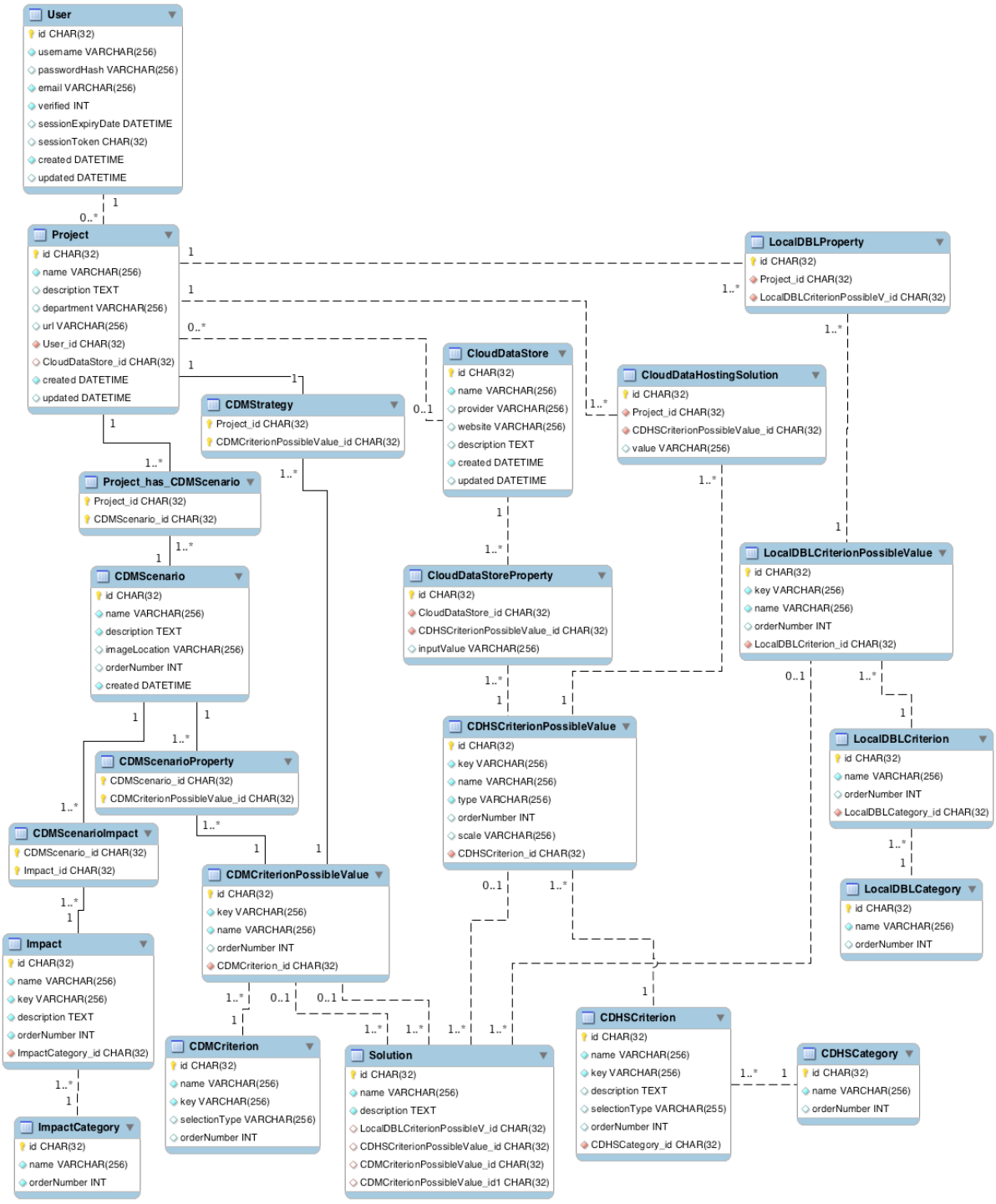

Abbildung 33 ER-Diagramm des Cloud Data Migration Tools

Abbildung 33 zeigt die Entitäten des Cloud Data Migration Tools. Das Diagramm wurde in der Data Modeling Perspektive der Anwendung MySQL Workbench erstellt und liegt auch dem Quellcode des Cloud Data Migration Tools bei. Ein Benutzer (Entität *User*) kann mehrere Projekte (Entität *Project*) haben, in denen er die Szenarien (Entität *Project_has_CDMScenario*) mit den zugehörigen Ausprägungen (Entität *CDMStrategy*) festlegt. Die Szenarien an sich sind in den Entitäten *CDMScenario, CDMScenario_has_CDMCriterionPossibleValue, CDMCriterionPossibleValue* und *CDMCriterion* beschrieben. Weiterhin kann die gewünschte Cloud Data Hosting Solution (Entität *CloudDataHostingSolution*) im Projekt beschrieben werden und ein Cloud Data Store (Entität

CloudDataStore) ausgewählt werden. Die Cloud Data Stores selbst sind ebenfalls beschrieben (Entitäten *CloudDataStoreProperty, CDHSCriterionPossibleValue, CDHSCriterion, CDHSCategory*). Neben der gewünschten Cloud Data Hosting Solution kann auch der aktuell genutzte DBL beschrieben werden (Entitäten *LocalDBLProperty, LocalDBLCriterionPossibleValue, LocalDBLCriterion, LocalDBLCategory*). Mögliche Konfliktlösungen aus Konflikten zwischen dem aktuell genutzten DBL, gewünschter Migrationsstrategie und der gewünschten Cloud Data Hosting Solution bzw. gewünschten Cloud Data Store sind u.a. in Form von Cloud Data Patterns in der Entität *Solution* beschrieben. Mögliche Auswirkungen der Datenmigration auf die Netzwerk-, Data Access oder Anwendungsebene werden in der Entität *Impact* beschrieben, über die Entität *ImpactCategory* gruppiert und mit der Entität *CDMScenarioImpact* den einzelnen Cloud Datenmigrationsszenarien zugeordnet.

Die oben genannten und abgebildeten Entitäten werden als Tabellen und zusätzlich eins zu eins als Java-Klassen umgesetzt und mit JDO Annotationen für die Persistenz sowie Gettern und Settern versehen. Alle Entitätsklassen erben von der abstrakten Klasse AbstractModel, welche Methoden zur Persistenz (save, delete, findAll, findById) beinhaltet. Sofern benötigt, werden die Entitäten weitere Methoden für Berechnungen (siehe Abschnitt 5.3.3) und spezielle Datenbankabfragen beinhalten.

Zu jedem Fragebogen und Formular (siehe Abschnitte 4.8 und 5.3.4) gibt es einen View (JSP) und einen Controller, der die Formulardaten entgegennimmt, validiert, in die entsprechenden Entitäten parst, persistiert und anschließend den Nutzer weiterleitet. Die Formulare innerhalb der Views werden dynamisch anhand der Beschreibungen der Kriterien mit ihren entsprechenden möglichen Werten automatisch generiert. Somit sind die Kriterienkataloge leicht erweiterbar.

Für die Migration werden keine Daten persistiert, da diese Daten (Verbindungsdaten, Passwörter, etc.) sehr sensibel sind. Es gibt zwei Interfaces SourceSystem (Methoden: getId(), getInstructions(), getConnectionParamters(), connect(HashMap<String, String> connectionProperties, ServletOutputStream out), getTablesAsCSV(ServletOutputStream out), getSQLCommands(ServletOutputStream out)) und TargetSystem (Methoden: getId(), getInstructions(), getConnectionParamters(), connect(HashMap<String, String> connectionProperties), migrate(String sqlCommands, HttpServletResponse resp) , migrate(HashMap<String, String> tableAsCsvData, HttpServletResponse resp), supportsSql()) welche von Adaptern für diverse Quell- und Zielsysteme implementiert werden können. Jeder Adapter erhält eine View für die Konfiguration und einen Controller, welcher SourceSystem oder TargetSystem implementiert und für die Verarbeitung der Formulardaten des zugehörigen Views verantwortlich ist. Der Prototyp ermöglicht die Migration zwischen SQL-fähigen Quell- und Zielsystemen als auch von einem SQL-fähigen Quellsystem zu einem NoSQL Zielsystem, über den Export der Daten und Spaltenbezeichner im CSV-Format.

5.3.3 Logik und Algorithmen

Der Decision Support Teil des Cloud Data Migration Tools soll laut FR-2 auch Konflikte ermitteln und Lösungen anbieten. Als Grundlage für die Konflikterkennung dienen dazu die Abbildungen der Szenarien auf die Migrationsstrategie (Taxonomie der Migrationsszenarien) aus Abschnitt 4.2, sowie die Abbildungen der Szenarien auf die Cloud Data Hosting Solutions aus Abschnitt 4.3.

Konfliktermittlung zwischen gewähltem Szenario und gewünschter Migrationsstrategie

Die Ermittlung der Konflikte wird wie folgt berechnet:

```
 1 // init
 2 numberOfScenarios = 11;
 3 numberOfCritera = 29;
 4 possibleCriteria[numberOfScenarios][numberOfCritera] = false;
 5 setPossibleCriteriaPerScenario(possibleCriteria); // according to chapter 4.2
 6 checkedScenarios[numberOfScenarios] = false;
 7 checkedCriteria[numberOfCritera] = false;
 8 setCheckedScenarios(checkedScenarios); // according to questionnaire
 9 setCheckedCriteria(checkedCriteria); // according to questionnaire
10 conflicts[numberOfScenarios][numberOfCritera] = false;
11 resolvedConflicts[numberOfCritera] = false;
12
13 // check for conflicts
14 for(scenario = 1 to numberOfScenarios){
15   for(criteria = 1 to numberOfCritera){
16     if(checkedScenarios[scenario] ==  true && checkedCriteria[criteria] == true && possible-
17 Criteria[scenario][criteria] == false){
18       conflicts[scenario][criteria] = true;
19     }
20   }
21 }
22
23 // check for automatically resolvable conflicts
24 for(scenario = 1 to numberOfScenarios){
25   for(criteria = 1 to numberOfCritera){
26     // a potential conflict
27     if(conflicts[scenario][criteria] == true){
28       // check if any other checked scenario can resolve this conflict
29       for(checkedScenario = 1 to numberOfScenarios){
30         if(checkedScenarios[checkedScenario] ==  true && possibleCrite-
31 ria[checkedScenario][criteria] == true){
32           resolvedConflicts[criteria] == true;
33         }
34       }
35     }
36   }
37 }
```

Listing 1 Konfliktermittlung zwischen Szenarien und möglichen Ausprägungen einer Cloud Datenmigration

Über eine doppelte For-Schleife (Zeile 14 und 15) wird für jedes ausgewählte Szenario jede Auswahl aus dem Migrationsstrategie Fragebogen mit der möglichen Ausprägung des entsprechenden Szenarios abgeglichen (Zeile 16 bis 17). Bei einem Konflikt wird die entsprechende Kombination aus Szenario und Kriterium markiert (Zeile 18). Sofern mehrere Szenarien ausgewählt wurden und Konflikte existieren wird geprüft ob die konfliktbehaftete Kombination aus Szenario und Kriterium in einem anderen ausgewählten Szenario erlaubt ist (Zeile 30 bis 33).

Ein Konflikt ist dabei eine Inkompatibilität eines Kriteriums mit einem Szenario. Sofern mehrere Szenarien ausgewählt wurden, kann ein Konflikt für ein Szenario durch ein anderes ausgewähltes Szenario gelöst werden. Beispielhaft seien die Szenarien „Reines Outsourcing des DBL“ und „Backup“, sowie das Kriterium „Kopie des DBL in die Cloud“ ausgewählt. Dieses Kriterium ist, wie man in Tabelle 1 sieht, inkompatibel zum Szenario „Reines Outsourcing des DBL“, aber kompatibel zum Szenario „Backup“, womit der Konflikt gelöst werden kann.

Konfliktermittlung zwischen gewähltem Szenario und gewünschter Cloud Data Hosting Solution

Die Ermittlung der Konflikte wird wie folgt berechnet:

```
38 // init
39 numberOfScenarios = 11;
40 numberOfCritera = 11;
41 possibleCriteria[numberOfScenarios][numberOfCritera] = false;
42 setPossibleCriteriaPerScenario(possibleCriteria); // according to chapter 4.3
43 checkedScenarios[numberOfScenarios] = false;
44 checkedCriteria[numberOfCritera] = false;
45 setCheckedScenarios(checkedScenarios); // according to questionnaire
46 setCheckedCriteria(checkedCriteria); // according to questionnaire
47 conflicts[numberOfScenarios][numberOfCritera] = false;
48
49 // check for conflicts
50 for(scenario = 1 to numberOfScenarios){
51   for(criteria = 1 to numberOfCritera){
52     if(checkedScenarios[scenario] ==  true && checkedCriteria[criteria] == true && possible-
53 Criteria[scenario][criteria] == false){
54       conflicts[scenario][criteria] = true;
55     }
56   }
57 }
```

Listing 2 Konfliktermittlung zwischen Szenarien und Cloud Data Hosting Solutions einer Cloud Datenmigration

Das Vorgehen ist dabei gleich zum obigen, jedoch wird die automatische Konfliktlösung ausgelassen, da die Ausprägungen einer Cloud Data Hosting Solution verschieden (distinct) und damit nicht kombinierbar sind.

Ein Konflikt ist in diesem Falle eine Inkompatibilität eines Cloud Data Hosting Solution Kriteriums mit einem Szenario. Diese Konflikte können nicht automatisch aufgelöst werden. Beispielhaft seien das Szenario „Reines Outsourcing des DBL“, sowie das Kriterium „SaaS“ ausgewählt. Dieses Kriterium ist, wie man in Tabelle 2 sieht, inkompatibel zum Szenario „Reines Outsourcing des DBL“, da eine SaaS Cloud Data Hosting Solution nicht genug Kontrolle oder Konfigurationsmöglichkeiten für ein reines Outsourcing des DBL anbieten.

Der Migrations-Teil des Cloud Data Migration Tools soll laut FR-3 verschiedene Zielsysteme unterstützen. Das Vorgehen des Datenexports und -imports wird dabei im Folgenden beschrieben.

MySQL Quellsystem

Als Quellsystem steht im Prototyp des Cloud Data Migration Tools MySQL zur Auswahl. Über das MySQL Client Kommandozeilenwerkzeug (mysql) wird versucht eine Verbindung mit den angegebenen Zugangsdaten zu einem MySQL Server herzustellen. Wenn dies erfolgreich war, wird das Kommandozeilenwerkzeug mysqldump verwendet um eine gesamte Datenbank sowohl im SQL Format, als auch im CSV Format zu exportieren.

Da der CSV Export von mysqldump lediglich die Werte, jedoch nicht die Spaltennamen enthält, müssen zusätzlich über die Metadatendatenbank von MySQL sämtliche Spaltennamen ermittelt und den kommagetrennten Werten im CSV Format vorangestellt werden.

Für den SQL Export ist der Kompatibilitätsmodus zu anderen Datenbanken zusätzlich einstellbar, welcher jedoch nur eine geringfügig verbesserte Kompatibilität bietet („`“ wird durch „‘“ ersetzt). Die jeweiligen Importadapter müssen also noch weitere Erset-

zungen und Umformungen vornehmen, damit der SQL Export von MySQL im jeweiligen Zielsystem interpretiert werden kann.

Da der MySQL Export Adapter die Kommandozeilentools mysql und mysqldump benötigt, kann das Cloud Data Migration Tool nicht in vollem Umfang in reinen Java PaaS Containern wie Google App Engine genutzt werden. Hier steht kein Zugriff auf das Dateisystem oder Möglichkeit der Installation externer Tools zur Verfügung.

Befehl zum Testen der Verbindungsdaten:
$> *PfadZuMySQLBinaries*/mysql -h *Servername* –P *Port* -u *Benutzername* -p *Passwort* -e Status *Datenbankname*

Befehl zum Exportieren der Struktur und Daten im (My)SQL Format:
$> *PfadZuMySQLBinaries*/mysqldump -h *Servername* –P *Port* -u *Benutzername* -p *Passwort* --hex-blob --compatible=*Kompatibilitätsmodus* --skip-extended-insert --skip-add-locks --skip-add-drop-table --skip-comments *Datenbankname*

Befehl zum Exportieren der Daten im CSV Format:
$> *PfadZurBash*/bash -c *PfadZuMySQLBinaries*/mysqldump -h *Servername* -P *Port* -u *Benutzername* -p *Passwort* -tab *PfadZuTemporärenVerzeichnis*/dump-*Datenbankname* *Datenbankname*

Befehl zum Exportieren der Spaltennamen:
$> *PfadZurBash*/bash -c *PfadZuMySQLBinaries*/mysql -h *Servername* -P *Port* -u *Benutzername* -p *Passwort Datenbankname* –e "select column_name into outfile *'PfadZuTemporärenVerzeichnis*/dump-*Datenbankname/Tabellenname-headings.csv*' from information_schema.columns where table_name='*Tabellenname*' AND table_schema='*Datenbankname*';"

Virtuelle Maschine Quellsystem
Um eine Abbild einer virtuellen Maschine zu exportieren, muss den Export Anweisungen der Virtualisierungssoftware gefolgt werden. Für einige Formate bieten Hersteller wie AWS für ihren EC2 Dienst auch direkte Anleitungen zum Import eines virtuellen Maschinen Abbilds. Daher ist es nicht ohne größeren Aufwand möglich, über das Cloud Data Migration Tool virtuelle Maschinen zu exportieren.

MySQL Zielsystem
Der Adapter für den Import von Struktur und Daten in eine MySQL Server Instanz nutzt den SQL Export des MySQL Quellsystems und importiert die Daten aus einer SQL Datei in eine existierende MySQL Datenbank.

Befehl für zum Importieren der Daten im SQL Format:
$> *PfadZurBash*/bash -c *PfadZuMySQLBinaries*/mysql -h *Servername* -P *Port* -u *Benutzername* -p *Passwort Datenbankname* < *PfadZuDateiMitSQLExport*

Oracle Zielsystem
Der Adapter für den Import von Struktur und Daten für Oracle Datenbanken (z.B. managed und hosted in Amazon RDS) nutzt ebenfalls den SQL Export des MySQL Quellsystems und passt die SQL Befehle wie folgt an:

- Escape-Sequenz “\'” wird durch “''” ersetzt

- Datentyp “int(11)” wird durch “int” ersetzt,
- Datentyp “text” wird durch “CLOB” ersetzt,
- Datentyp “datetime” wird durch “DATE” ersetzt,
- „ON DELETE NO ACTION ON UPDATE NO ACTION” wird entfernt,
- einfache “KEY” Definitionen werden entfernt,
- “NOT NULL DEFAULT *” wird durch “DEFAULT * NOT NULL“ ersetzt,
- “UNIQUE KEY” Definitionen warden entfernt,
- Fremdschlüssel Definitionen werden an das Ende des SQL Skriptes verschoben und
- Werte des Datentyps „datetime” (Format „yyyy-mm-dd hh24:mi:ss“) werden mittels der TO_DATE Funktion importiert.

SQL Server Zielsystem
Der Adapter für den Import von Struktur und Daten für Microsoft SQL Server Datenbanken (z.B. gemanagt und hosted in Amazon RDS) nutzt ebenfalls den SQL Export des MySQL Quellsystems und passt die SQL Befehle wie folgt an:

- Escape-Sequenz “\'” wird durch “''” ersetzt
- Datentyp “int(11)” wird durch “int” ersetzt,
- einfache “KEY” Definitionen werden entfernt,
- “UNIQUE KEY” Definitionen werden entfernt und
- Fremdschlüssel Definitionen werden an das Ende des SQL Skriptes verschoben.

Google Cloud SQL Zielsystem
Obwohl Google Cloud SQL zwar ein managed und hosted MySQL Server ist, ist es nicht möglich sich mit dem Kommandozeilentool mysql auf seiner MySQL Serverinstanz einzuloggen. Google bietet aber eine Importanleitung [152] an, welche verlangt, dass zuerst ein Dump der bestehenden MySQL Datenbank erstellt wird, dieser in ein Bucket eines Google Cloud Storage Accounts hochgeladen wird und anschließend über die Google APIs Konsole unter Angabe des Bucket- und Dateinamens importiert werden soll.

Der Google Cloud SQL Adapter bietet den Dump der MySQL Datenbank zum Download an und fordert den Anwender auf den weiteren Schritten von Googles Anleitung zu folgen. Alternativ wäre es auch in einer zukünftigen Version möglich, die SQL Befehle des Dumps der Reihe nach über JDBC an die Google Cloud SQL Instanz zu schicken.

Virtuelle Maschine Zielsystem
Um eine Abbild einer virtuellen Maschine zu importieren, muss den Import Anweisungen (z.B. Amazon VM Import/Export Dienst [153]) des jeweiligen Cloud Dienstanbieters gefolgt werden. Die Abbilder müssen in der Regel in ein proprietäres Format des Cloudanbieters konvertiert, anschließend hochgeladen und konfiguriert werden. Daher ist es nicht ohne größeren Aufwand möglich, über das Cloud Data Migration Tool virtuelle Maschinen zu importieren.

Amazon SimpleDB Zielsystem
Der Amazon SimpleDB Adapter nutzt den CSV Export des MySQL Quelladapters. Für jede Tabelle wir eine „Domain“ innerhalb SimpleDB mit einem Präfix (typischerweise der Datenbankname) angelegt und deren Datensätze in Form von Key-Values Paaren importiert. Dabei wird angenommen, dass die erste Spalte dem Primärschlüssel entspricht, zusammengesetzte Schlüsselattribute werden also noch nicht unterstützt. Weiterhin

schlägt der Import fehl, falls ein Datensatz ein Amazon SimpleDB Limit [98] überschreitet (z.B. ein Wert einer Spalte größer als 1KB ist).

5.3.4 User Interface und GUI Prototyp

Die folgenden Abbildungen stellen Teile der graphischen Benutzeroberflächen dar und zeigen die Interaktionsmöglichkeiten mit dem Cloud Data Migration Tool auf. Auf eine vollständige Entwicklung der GUI als GUI Prototyp wird verzichtet, da die entsprechenden Formulare aus den Werten die in der Datenbank gespeichert sind, generiert werden.

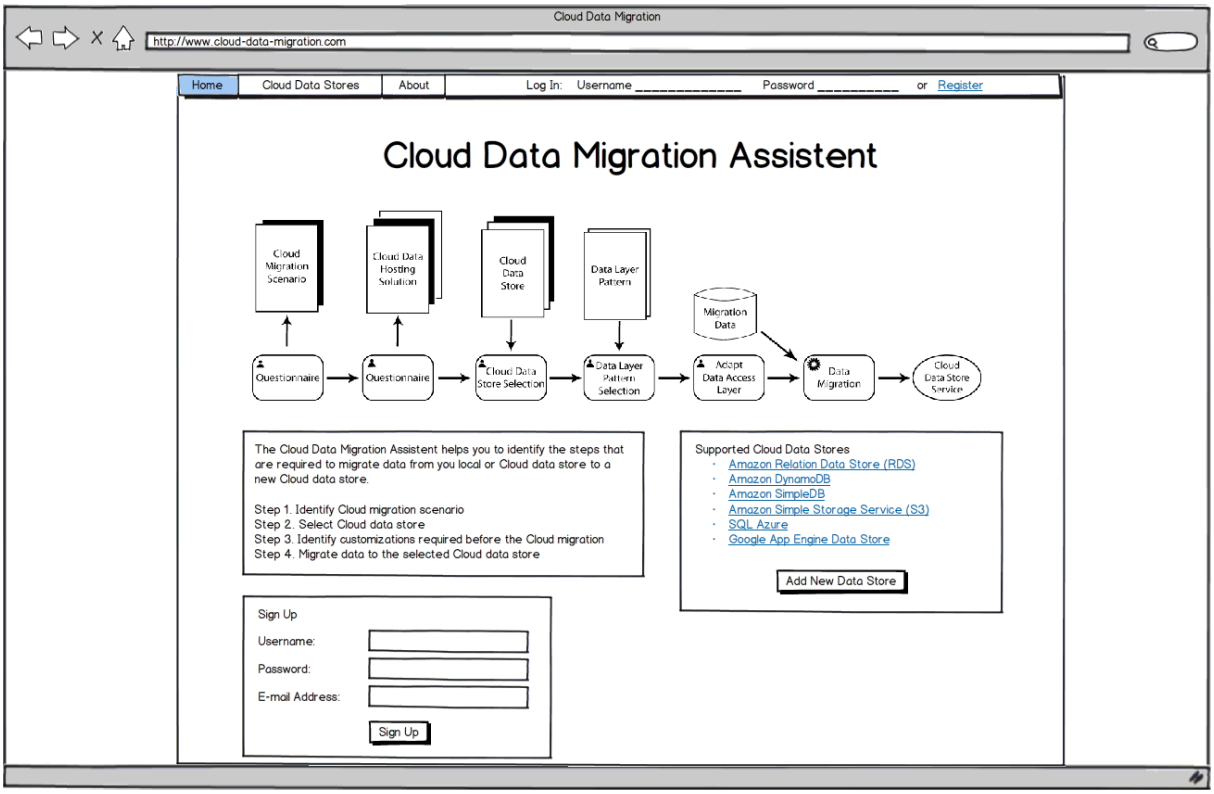

Abbildung 34 Startseite des Cloud Data Migration Tools

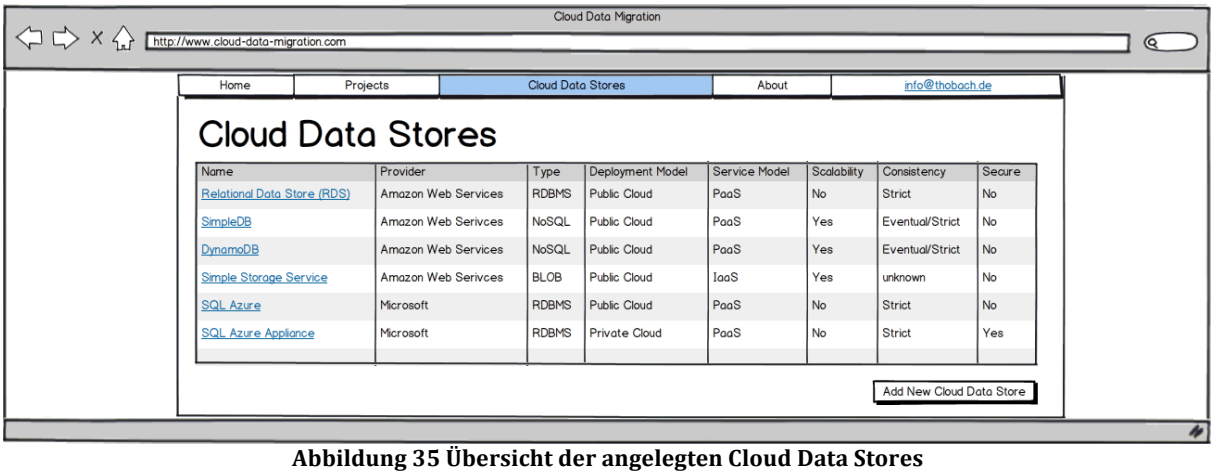

Abbildung 35 Übersicht der angelegten Cloud Data Stores

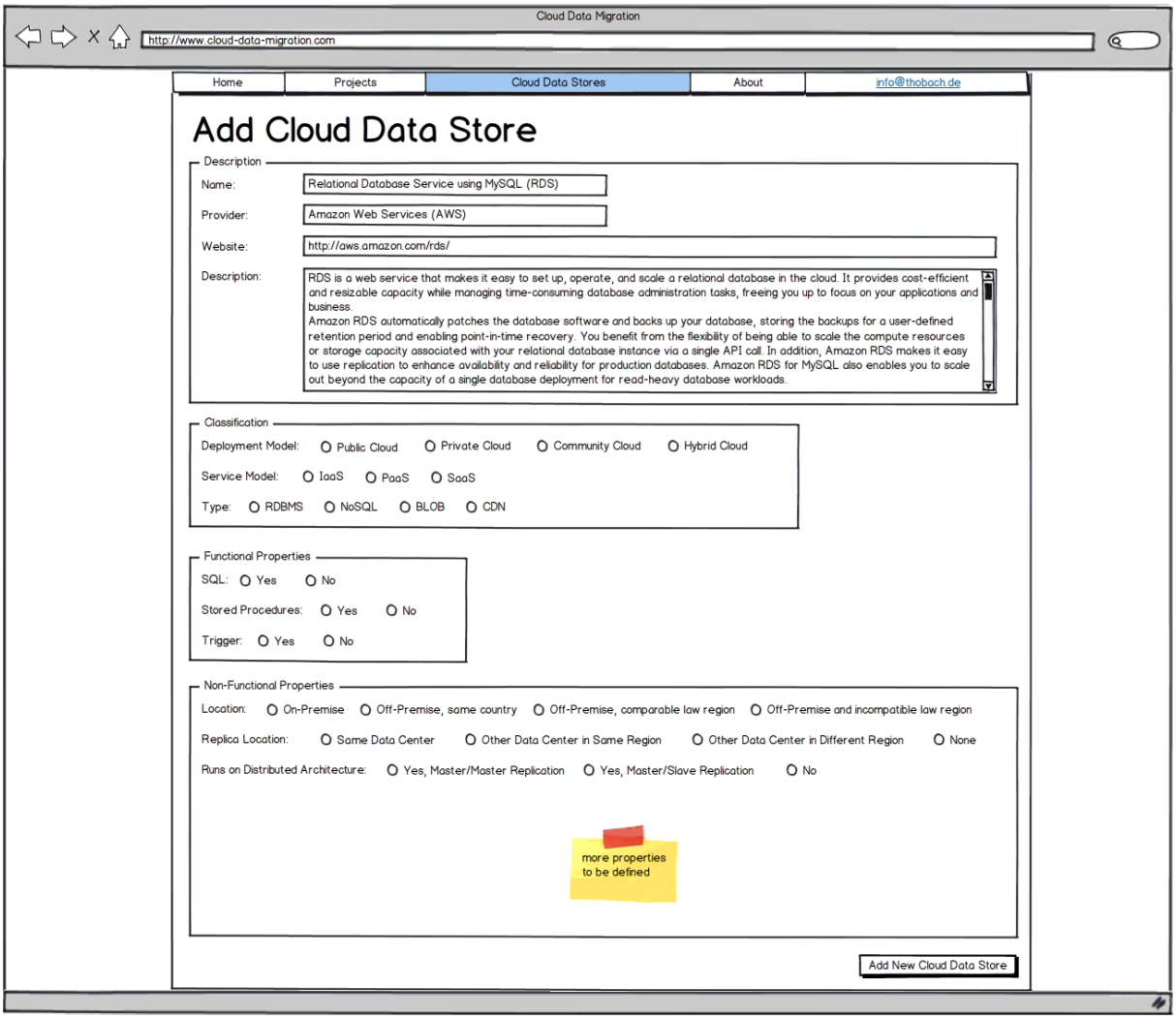

Abbildung 36 Formular zum Hinzufügen von neuen Cloud Data Stores

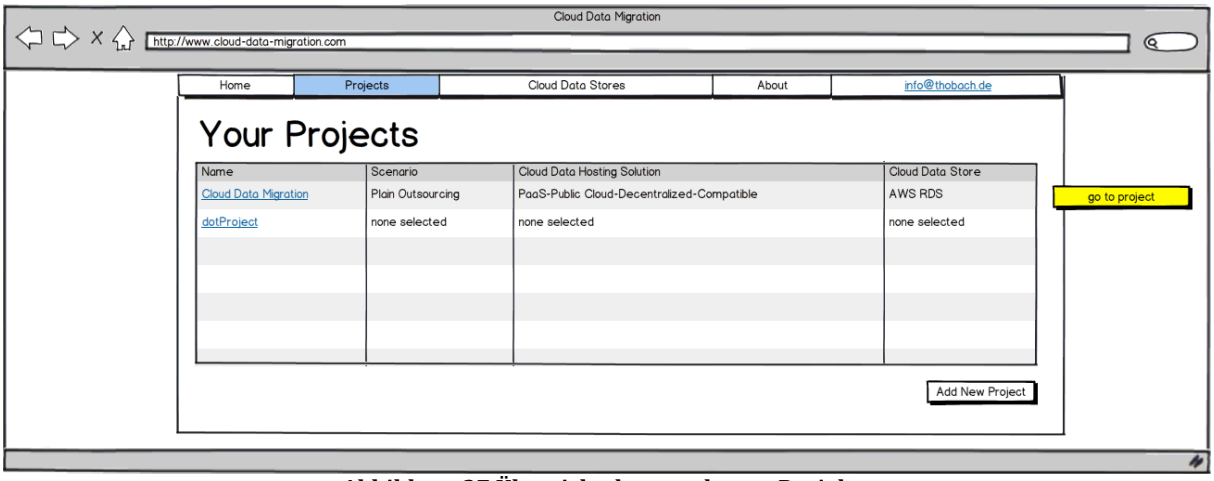

Abbildung 37 Übersicht der angelegten Projekte

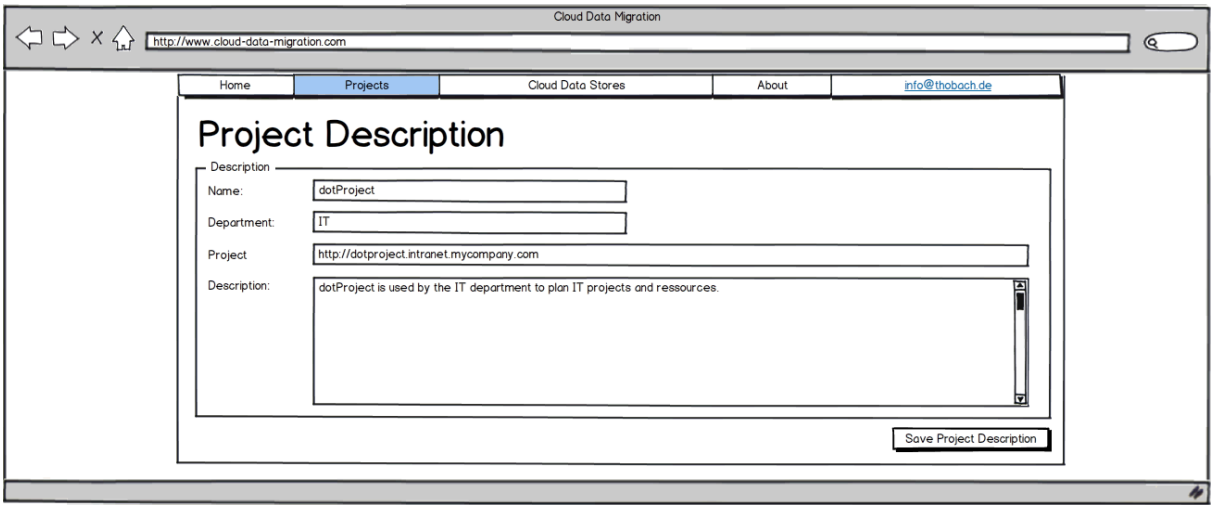

Abbildung 38 Formular zum Hinzufügen von neuen Projekten

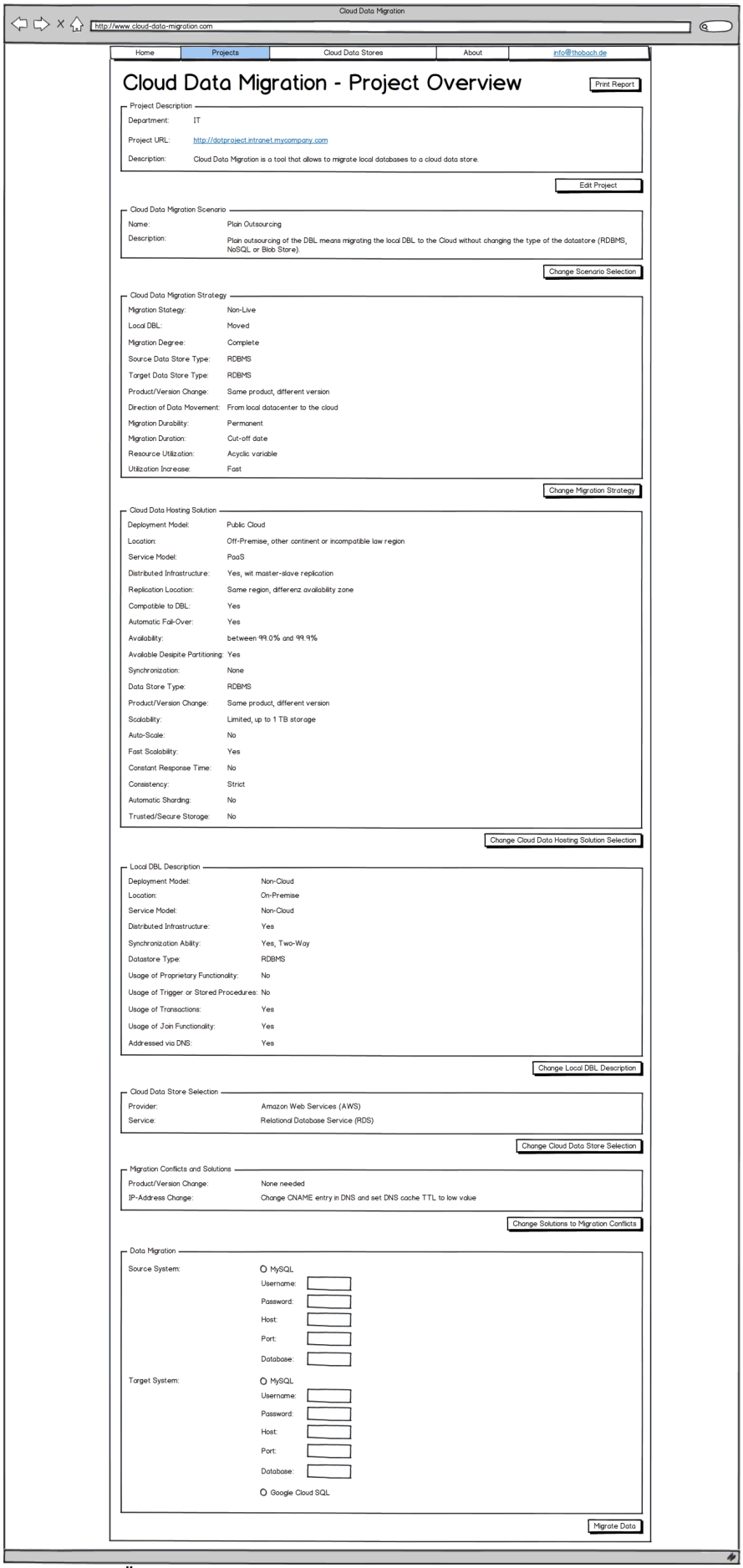

Abbildung 39 Überblick über getroffene Entscheidungen eines Migrationsprojekts

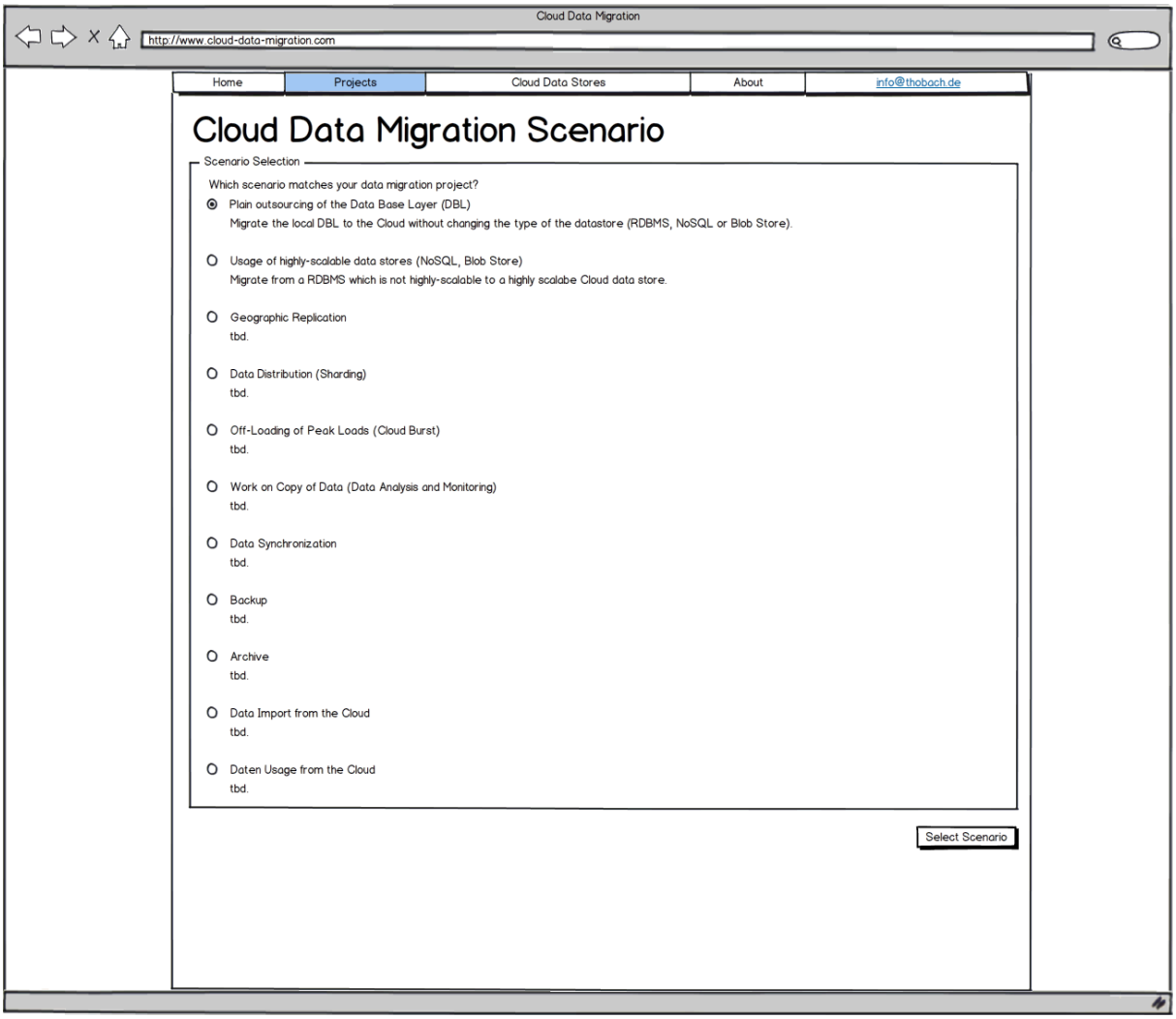

Abbildung 40 Formular zur Auswahl eines Cloud Datenmigrationsszenarios

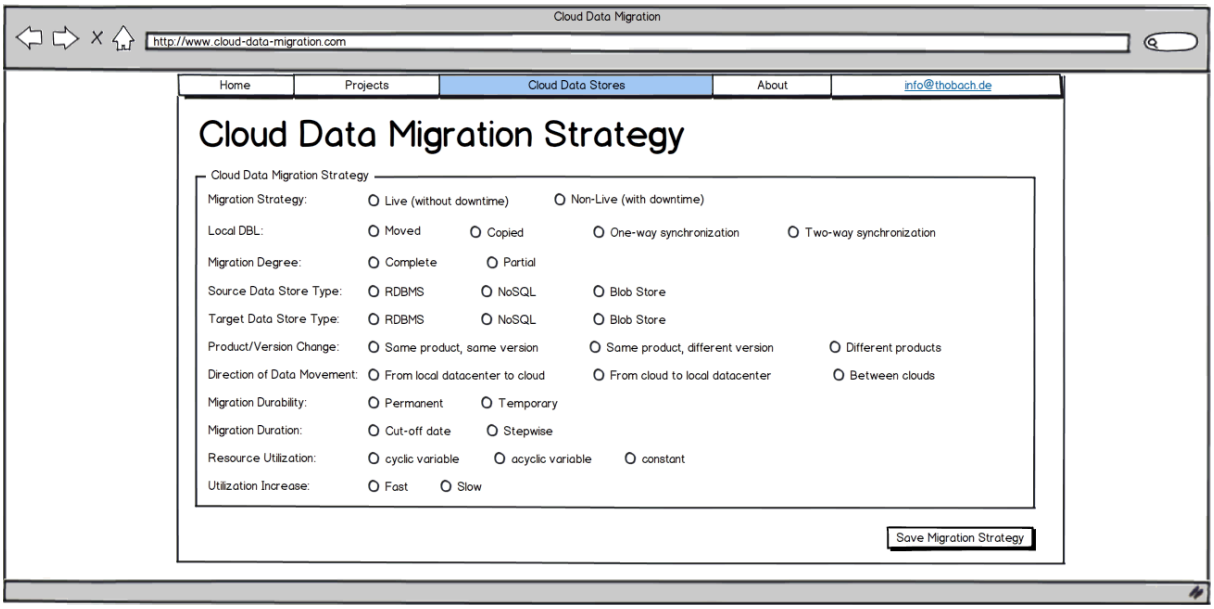

Abbildung 41 Formular zur Auswahl einer Cloud Datenmigrationsstrategie

5.4 Prototypische Implementierung

Dieser Abschnitt beschreibt wie die Implementierung eines Prototyps für das Cloud Data Migration Tool ablief und welche Schwierigkeiten es gab.

5.4.1 Entwicklungsumgebung

Die Implementierung erfolgte in der IDE Eclipse Indigo JEE. Die Artefakte wurden in einem öffentlichen GIT Repository auf github hinterlegt: https://github.com/thobach/cloud-data-migration und die einzelnen Entwicklungsaufgaben in JIRA hinterlegt und mit jedem Commit verknüpft.

5.4.2 Schwierigkeiten bei der Implementierung

Classification Modul

Die Anwendung wurde zuerst für eine lokale MySQL Datenbank ausgelegt, was ohne Probleme möglich war. Danach wurde sie erweitert um zusätzlich auch in der Google App Engine mit dem Google Cloud SQL Dienst zusammenzuarbeiten. Dabei gab es Probleme durch inkompatible Bibliotheksversionen (JDO und Datanucleus) sowie schlecht dokumentierten JDO Konfigurationsparametern.

Migration Modul

Beim Adapter für das Google App Engine Datastore Zielsystem wurde zunächst versucht das Google Tool Bulk Loader zu verwenden um fertige Konfigurationstemplates für den Datenimport zu generieren. Allerdings unterstützt dieses Tool nur den veralteten, und für neue Projekte nicht mehr angebotenen, Master/Slave Datastore und nicht den aktuellen High Replication Datastore. Daher müssen Konfigurationsdateien nun manuell erstellt werden und ein automatisierter Import der Daten in den Google App Engine Datastore ist nicht möglich. Alternativ wäre es auch möglich gewesen ähnlich zum SimpleDB Zieladapter die Daten per Low-Level Datastore API zu importieren. Hierzu hätten aber auch die Fremdschlüsselbeziehungen angegeben werden müssen, welche aus CSV Dateien nicht ableitbar sind und somit einen größeren Aufwand beim Export verlangt hätten.

Für den Adapter des Oracle Zielsystems wurde zunächst versucht den Import über das Kommandozeilenprogramm sqlplus zu erledigen. Allerdings hat dieses Tool einige zusätzliche Funktionen wie Variablenerkennung innerhalb von SQL Kommandos durch Voranstellung des &-Symbols, was den Import erschwerte, da diese Funktion temporär deaktiviert werden musste. Außerdem hätte dieses Tool eine weitere Abhängigkeit mit sich gebracht und wurde somit durch die Oracle JDBC Schnittstelle abgelöst. Die SQL Befehle zum Erstellen des Schemas und Import der Daten werden nun in einer Transaktion über JDBC auf dem entfernten Oracle Server ausgeführt. Das gleiche Verfahren wurde auch für Microsofts SQL Server angewandt. Beim MySQL Zieladapter besteht schon durch den Quelladapter eine Abhängigkeit zu den MySQL Kommandozeilentools, daher wird hier weiter das „mysql“ Tool für den Datenimport verwendet.

6 Anwendung der Arbeitsergebnisse

Um die Funktionsweise des Cloud Data Migration Tools zu überprüfen soll die gesamte Anwendung selbst in die Cloud migriert werden, und damit auch der DBL des Cloud Data Migration Tools.

6.1 Beispielhafte Anwendungsfälle

Im ersten Szenario wird der Fall „Datennutzung aus der Cloud“ untersucht. Dabei wird der DBL sowohl zu Google Cloud SQL, als auch Amazon EC2 mit MySQL Server und Azure SQL migriert. Die lokale Anwendung soll damit weiter funktionieren. Anschließend soll die gesamte Anwendung in die AWS Cloud migriert werden (Szenario 2: Reines Outsourcing des DBL) und der DBL auf Amazon EC2 mit MySQL Server genutzt werden. Zuletzt wird versucht die Daten des Cloud Data Migration Tools in einen NoSQL Speicher zu importieren und einen Zugriff der Anwendung auf die Daten in diesem NoSQL Speicher zu ermöglichen (Szenario 3: Verwendung von hochskalierbaren Datenspeicher).

6.2 Protokoll der Cloud Daten Migration

Szenario 1.1: Datennutzung aus der Cloud (Google Cloud SQL)

Anhang 1 zeigt die getroffenen Auswahlen und Konflikte bzw. Hinweise des Cloud Data Migration Tools für die Migration des DBL des Cloud Data Migration Tools. In diesem Fall wurde der Google Cloud SQL Dienst als Cloud Datastore ausgewählt.

Das Cloud Data Migration Tool identifizierte dabei folgende Konflikte bei der Auswahl des Google Cloud SQL Dienstes:

- Scalability - Degree of Automation: Partial Automation (Mix of Aut. and Manual) - Google Cloud SQL has: Manual
- Security - Storage Encryption: Yes - Google Cloud SQL has: No
- Security - Firewall: Yes - Google Cloud SQL has: No
- Security - Confidentiality: Yes - Google Cloud SQL has: No
- Security - Integrity: Yes - Google Cloud SQL has: No

Die nachfolgenden Konflikte wurden zwar zusätzlich aufgelistet, konnten aber jeweils durch eine andere mögliche Auswahl bei der Spezifikation der Cloud Data Hosting Solution kompensiert werden:

- Interoperability - Developer SDKs: Java - Google Cloud SQL has: None Needed (Standard Support)
- Cloud Computing - Deployment Model: Private Cloud - Google Cloud SQL has: Public Cloud
- Cloud Computing - Deployment Model: Community Cloud - Google Cloud SQL has: Public Cloud
- Cloud Computing - Deployment Model: Hybrid Cloud - Google Cloud SQL has: Public Cloud

Von den 22 identifizierten Patterns und Empfehlungen kamen generell drei (#8, #14 und #21) nicht in Frage.

Von den 16 identifizierten Anpassungsmöglichkeiten des DAL bzw. der Anwendungsschicht kamen generell 12 nicht in Frage. Diese geringe Relevanz ist dadurch zu erklären, dass nur die Auswahl des Migrationsszenarios Auswirkungen auf die Anpassungsmöglichkeiten hat und nicht die detaillierteren Auswahlen der Migrationsstrategie, Cloud Data Hosting Solution und des lokalen DBL.

Damit das Cloud Data Migration Tool die Cloud Data Hosting Solution Google Cloud SQL nutzen konnte, musste die Anwendung wie folgt angepasst werden:

- Downgrade der JDO und Datanucleus Libraries
- Verwendung der Google Cloud SQL JDBC Treiber (com.google.cloud.sql.Driver)
- Anpassung der Connection Strings (jdbc:google:rdbms://project-id:instance-id/datenbankname)
- Explizite Spezifikation der Spaltennamen im Falle von Bezeichnern die aus mehreren Worten zusammengesetzt wurden (Camel Case) anhand der „column" Eigenschaft der @Persistent Annotation, auf Grund der Unterscheidung von Groß- und Kleinschreibung
- Explizite Spezifikation der Tabellennamen anhand der „table" Eigenschaft der @PersistenceCapable Annotation, auf Grund der Unterscheidung von Groß- und Kleinschreibung.

Szenario 1.2: Datennutzung aus der Cloud (Amazon EC2 MySQL Server - IaaS)
Anhang 2 zeigt die getroffenen Auswahlen und Konflikte bzw. Hinweise des Cloud Data Migration Tools für die Migration des DBL des Cloud Data Migration Tools, die unterschiedlich zum Szenario 1.1 sind. In diesem Fall wurde der Amazon EC2 Dienst mit einer Installation des MySQL Servers als Cloud Datastore ausgewählt.

Das Cloud Data Migration Tool identifizierte dabei folgende Konflikte bei der Auswahl des Amazon EC2 Dienstes:

- Scalability - Degree of Automation: Partial Automation (Mix of Aut. and Manual) - Amazon EC2 has: Manual
- Management and Maintenance effort - Degree of Automation: Managed/Automated - Amazon EC2 has: Self-Service

Von den sieben identifizierten Patterns und Empfehlungen kam generell eins (#5) nicht in Frage.

Von den 16 identifizierten Anpassungsmöglichkeiten des DAL bzw. des Anwendungsschicht kamen generell 12 nicht in Frage. Der Grund für die geringe Relevanz ist der gleiche wie im Szenario 1.1.

Damit das Cloud Data Migration Tool die Cloud Data Hosting Solution Amazon EC2 mit MySQL Server nutzen konnte, musste die Anwendung wie folgt angepasst werden:

- Anpassung der Connection Strings (jdbc:mysql://ec2-server-name:3306/datenbankname)
- Da das VM Import/Export Tool nur Windows Server 2003/2008 Images importieren kann, musste eine Windows Server Installation für den MySQL Server verwendet werden. Als Format gab das VM Import Tool an, dass VHD, VMDK und RAW unterstützt werden, allerdings konnte das VMDK Format von Virtual Box nicht importiert werden, auf Grund eines inkompatiblen Formats („ERROR: File uses unsupported compression algorithm 0"). Nach einer Umwandung des VMDK Formates in das RAW Format mit „$> VBoxManage internalcommands convertoraw MySQL-Server-Windows-Server-2008-R2-SP1.vmdk mysql.img" ließ sich der Import des 12 GB große Image mit „$> ec2-import-instance mysql.img -t m1.small -f RAW -a x86_64 -b *s3bucketname* -o *accesskey* -w *secretaccesskey*" starten.

Da nach ca. 20% des Uploads das System des Autors abstürzte und der Festplattenspeicher zu knapp wurde um einen weiteren Versuch zu starten, wurde das Szenario abgebrochen.

Szenario 1.3: Datennutzung aus der Cloud (Azure SQL, Produktwechsel)
Anhang 3 zeigt die getroffenen Auswahlen und Konflikte bzw. Hinweise des Cloud Data Migration Tools für die Migration des DBL des Cloud Data Migration Tools, die unterschiedlich zu Szenario 1.1 sind. In diesem Fall wurde der Azure SQL Dienst als Cloud Datastore ausgewählt.

Das Cloud Data Migration Tool identifizierte dabei folgende Konflikte bei der Auswahl des SQL Database Dienstes:

- Security - Storage Encryption: Yes - SQL Database (Windows Azure) has: No
- Security - Confidentiality: Yes - SQL Database (Windows Azure) has: No
- Security - Integrity: Yes - SQL Database (Windows Azure) has: No
- Backup - Backup Method: Snapshot (file system) - SQL Database (Windows Azure) has: File-backup (export)

Der nachfolgende Konflikt wurde zwar zusätzlich aufgelistet, konnten aber durch eine andere mögliche Auswahl bei der Spezifikation der Cloud Data Hosting Solution kompensiert werden:

- Interoperability - Developer SDKs: Java - SQL Database (Windows Azure) has: .Net, None Needed (Standard Support)

Von den 22 identifizierten Patterns und Empfehlungen kam generell eins (#14) nicht in Frage.

Von den 16 identifizierten Anpassungsmöglichkeiten des DAL bzw. des Anwendungsschicht kamen acht generell nicht in Frage. Der Grund für die geringe Relevanz ist der gleiche wie im Szenario 1.1.

Damit das Cloud Data Migration Tool die Cloud Data Hosting Solution SQL Database nutzen konnte, musste die Anwendung wie folgt angepasst werden:

- Anpassung der Connection Strings (jdbc:sqlserver://sql-database-server-name:1433;database=datenbankname;encrypt=true;hostNameInCertificate=*.database.windows.net;loginTimeout=30;)

Szenario 2.1: Reines Outsourcing des DBL
Anhang 4 zeigt die getroffenen Auswahlen und Konflikte bzw. Hinweise des Cloud Data Migration Tools für die Migration des DBL des Cloud Data Migration Tools, die unterschiedlich zu Szenario 1.1 sind. In diesem Fall wurde der Amazon EC2 Dienst mit einem Server Image mit MySQL Server als Cloud Datastore ausgewählt.

Das Cloud Data Migration Tool identifizierte dabei folgende Konflikte bei der Auswahl des Amazon EC2 Dienstes:

- Scalability - Degree of Automation: Partial Automation (Mix of Aut. and Manual) - Amazon EC2 has: Manual
- Management and Maintenance effort - Degree of Automation: Managed/Automated - Amazon EC2 has: Self-Service

Die nachfolgenden Konflikte wurden zwar zusätzlich aufgelistet, konnten aber durch eine andere mögliche Auswahl bei der Spezifikation der Cloud Data Hosting Solution kompensiert werden:

- Cloud Computing - Deployment Model: Private Cloud - Amazon EC2 has: Public Cloud
- Cloud Computing - Deployment Model: Community Cloud - Amazon EC2 has: Public Cloud
- Cloud Computing - Deployment Model: Hybrid Cloud - Amazon EC2 has: Public Cloud

Von den sieben identifizierten Patterns und Empfehlungen kam generell eins (#5) nicht in Frage.

Von den 16 identifizierten Anpassungsmöglichkeiten des DAL bzw. des Anwendungsschicht kamen generell 12 nicht in Frage. Der Grund für die geringe Relevanz ist der gleiche wie im Szenario 1.1.

Der Migrationsversuch musste auf Grund des vorzeitigen Scheiterns von Szenario 1.2 abgebrochen werden, da kein Image einer lokalen virtuellen Maschine in den EC2 Dienst importiert werden konnte.

Szenario 3.1: Verwendung von Hochskalierbaren Datenspeichern (NoSQL, BLOB Datenspeicher)

Anhang 5 zeigt die getroffenen Auswahlen und Konflikte bzw. Hinweise des Cloud Data Migration Tools für die Migration des DBL des Cloud Data Migration Tools, die unterschiedlich zu Szenario 1.1 sind. In diesem Fall wurde der Amazon SimpleDB Cloud Datastore ausgewählt.

Das Cloud Data Migration Tool identifizierte dabei folgende Konflikte bei der Auswahl des Amazon SimpleDB Dienstes:

- Scalability - Degree: Virtually Unlimited - Amazon SimpleDB has: Limited (up to 10 GB per domain, up to 1 billion attributes per domain)
- Security - Storage Encryption: Yes - Amazon SimpleDB has: No
- Security - Firewall: Yes - Amazon SimpleDB has: No
- Security - Confidentiality: Yes - Amazon SimpleDB has: No
- Security - Integrity: Yes - Amazon SimpleDB has: No
- Interoperability - Data Portability: Import - Amazon SimpleDB has: None
- Interoperability - Data Portability: Export - Amazon SimpleDB has: None
- Performance - Predictability Read/Write/Response: Predictable, x milliseconds (10) - Amazon SimpleDB has: Non-Predictable
- Backup - Backup Interval: On Demand - Amazon SimpleDB has: None
- Backup - Number of Backups Kept: Number of backups (30) - Amazon SimpleDB has: None

Die nachfolgenden Konflikte wurden zwar zusätzlich aufgelistet, konnten aber durch eine andere mögliche Auswahl bei der Spezifikation der Cloud Data Hosting Solution kompensiert werden:

- Interoperability - ORM: JDO - Amazon SimpleDB has: JPA
- Cloud Computing - Deployment Model: Private Cloud - Amazon SimpleDB has: Public Cloud

- Cloud Computing - Deployment Model: Community Cloud - Amazon SimpleDB has: Public Cloud
- Cloud Computing - Deployment Model: Hybrid Cloud - Amazon SimpleDB has: Public Cloud

Von den 28 identifizierten Patterns und Empfehlungen kamen generell sieben (#1, #4, #11, #17, #18, #22, #24) nicht in Frage. Der Grund für sieben unrelevanten Treffer ist, dass Texteingaben bei der Beschreibung des Cloud Data Stores nicht verglichen werden können (#1, #17, #22), Angaben des lokalen DBL nicht beachtet werden (#4, #18, #24) und zur Verwendung des Confidentiality Level Aggregator Pattern (#11) keine spezifischen Fragen existieren und es somit für alle Szenarien empfohlen wird.

Von den 17 identifizierten Anpassungsmöglichkeiten des DAL bzw. des Anwendungsschicht kamen drei generell nicht in Frage. Der Grund für die drei unrelevanten ist der gleiche wie im Szenario 1.1.

Damit das Cloud Data Migration Tool die Cloud Data Hosting Solution Amazon SimpleDB nutzen konnte, musste die Anwendung wie folgt angepasst werden:

- Der Import verlief ohne Probleme, abgesehen von Tabellen mit zusammengesetzten Primärschlüsseln, hier müsste der Import-Adapter angepasst werden um die zusammengesetzten Schlüssel zu konkatenieren und als einen Schlüssel verwenden, oder einen künstlichen neuen Primärschlüssel einführen.
- Der Versuch die aktuelle Version vom Februar 2012 von SimpleJPA als ORM zu verwenden, scheiterte an bekannten Bugs (einige Funktionen sind inkompatibel mit Mac OS X). Die vorherige Version 1.5 verlangte Offsets bei Integer Werten, obwohl das Cloud Data Migration Tool keine negativen Werte verwendet. Die Nutzung von Funktionen, die keine Entitäten mit Integer Werten verwendeten, war jedoch möglich.

Auf Grund des fehlgeschlagenen Versuchs der Nutzung von Amazon SimpleDB unter Nutzung des ORM SimpleJPA wurde versucht die Daten in einen weiteren NoSQL Datenspeicher (Google App Engine High Replication Datastore) zu importieren und von da aus zu nutzen.

Der Import der Daten in den Google App Engine High Replication Datastore verlief mit der manuell erstellten Migrationskonfiguration und dem Bulk Loader Tool von Google ohne Probleme. Der Versuch die JDO Implementierung von Datanucleus für den App Engine Datastore zu verwenden ist allerdings an der aufwändigen Konvertierung aller Primär- und Fremdschlüssel vorerst auf Grund Zeitmangels abgebrochen worden.

6.3 Evaluierung der Ergebnisse

Durch die textuelle Beschreibung des Migrationsprojektes, die Auswahl der Szenarien und Definition der Migrationsstrategie konnte der Rahmen der Migration definiert werden. Die meisten Auswahlen in diesem Abschnitt hatten aber bis auf die Inkompatibilitäten mit der Migrationsstrategie keine weiteren Auswirkungen auf Patterns, Konflikte oder Empfehlungen.

Die darauf folgende Beschreibung der Anforderungen an eine Cloud Data Hosting Solution war hilfreich um sich bewusst zu machen, auf welche technischen Kriterien es an-

kommt und welche Möglichkeiten bestehen. Bei einer Mehrfachauswahl innerhalb der Spezifikation der Cloud Data Hosting Solution wäre es hilfreich zusätzlich definieren zu können, ob alle Auswahlen Pflicht sind, oder ob eine der Auswahlen ausreicht. Dadurch hätte z.B. bei der Spezifikation des Deployment Models angegeben werden können, dass sowohl Private Cloud, Public Cloud, Hybrid Cloud und Community Cloud in Frage kommen und nicht alle Deployment Modelle unterstützt werden müssen. Bei der Konfliktauswertung in der Auflistung der passenden Cloud Data Stores hätten somit drei potentielle Konflikte weniger bestanden. Gleiches trifft auf Developer SDKs zu, wo Java und None Needed (Standard Support) ausgewählt wurden, Google Cloud SQL aber nur None Needed (Standard Support) anbietet und somit das „fehlende" Java SDK eigentlich kein Konflikt ist.

Die Auswahl eines passenden Cloud Data Stores war in den meisten Fällen noch mühsam, da keine KO-Kriterien bzw. Priorisierungen der Cloud Data Hosting Solution Kriterien vorlag. Hilfreiche wäre es z.B. nach Service Model oder Produkt-Kompatibilität zu filtern.

Bei der Beschreibung der aktuellen, lokalen Datenschicht war die Frage nach dem lokalen Datenspeichertyp eigentlich nicht notwendig, wenn der Datenspeichertyp des Quellsystems schon bei der Migrationsstrategie angegeben wurde. Ansonsten ist aus diesem Fragebogen auch nur noch die Frage nach der aktuellen Adressierung der Datenbankserver notwendig um potentielle Konflikte aufzuzeigen bzw. Pattern Vorschläge oder anderweitige Migrationsempfehlungen zu geben. Alle weiteren Fragen dienen nur dem Bewusst werden der Unterschiede zwischen der aktuellen und zukünftigen Datenschicht. In Zukunft können die Angaben aber auch genutzt werden um weitere Konflikte, Patterns oder Hilfestellungen für die Migration zu geben oder die Vorschläge zu Patterns und Anpassungen an höhere Anwendungsschichten präziser zu gestalten (siehe Szenarien 1.1 und 3.1).

Die darauf folgenden Vorschläge zu Cloud Data Patterns und Empfehlungen bzgl. Skalierbarkeit, Vertraulichkeit und Anpassungsmöglichkeiten höherer Schichten waren insgesamt sehr hilfreich und enthielten wenig irrelevante Aspekte.

Hingegen waren bei den spezifischen Anpassungsmöglichkeiten auf Netzwerk-, Data Access- und Anwendungsebene i.d.R. weit weniger als die Hälfte der Vorschläge relevant. Es gab auch eine Dopplung bzgl. der Adressierbarkeit des neuen Zielspeichers mit den vorherigen Empfehlungen. Grundsätzlich sollten die Vorschläge zu den Anpassungsmöglichkeiten nicht nur auf der Auswahl des Migrationsszenarios, sondern auch auf der Auswahl der Migrationsstrategie, des Cloud Data Stores und der Beschreibung des lokalen DBL beruhen.

Die eigentliche Datenmigration zwischen RDBMS bzw. einem RDBMS und den angebotenen NoSQL Speichern verlief ohne Probleme. Allerdings wurde sie auch intensiv zuvor mit der lokalen Datenbank des Cloud Data Migration Tools getestet. Es ist abzuwarten, inwiefern die Migration der Datenschicht anderer Anwendungen auch erfolgreich ist.

Bei der Anpassung des DAL nach einem Wechsel von einem RDBMS zu einem NoSQL Speicher ist die Nutzung der migrierten Daten aufwändiger. Es gibt zwar z.B. für Amazon SimpleDB eine JPA Implementierung welche als ORM eingesetzt werden kann, jedoch ist diese noch instabil und erfordert eine aufwändigere Datenkonvertierung z.B. bei Zahlen

auf Grund von lexikographischer Sortierung oder bei (zusammengesetzten) Primärschlüsseln. Ähnliche Probleme gab es auch bei der Nutzung des Google App Engine Datastores.

Neben diesen funktionalen Eigenschaften lag die Ladezeit einzelner Seiten innerhalb des Cloud Data Migration Tools zum Teil über zwei Sekunden, was bei der Nutzung hinderlich ist, wenn man mehrere Auswahlen testen möchte um Konflikte zu lösen. Ein besseres Caching und optimierte Abfragen können helfen diese Wartezeit zu reduzieren.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

Diese Arbeit stellt eine ganzheitliche und erweiterbare Migrationsmethodik vor, welche auf der Migrationsmethodik von Laszewski [17] aufbaut und diese um neun zusätzliche Migrationsszenarien und drei weitere Quell- und Zieldatenspeichertypen (NoSQL, BLOB Datenspeicher und CDN) erweitert.

Die Cloud Data Hosting Taxonomie von Strauch [11] wurde von fünf auf über 50 anwendbare Kriterien erweitert, um Cloud Datenspeicher genauer zu beschreiben, Cloud Data Patterns [12] zuzuordnen und weitere Hilfestellungen bei der Daten-, Datenschicht- und Anwendungsmigration zu geben.

Mit dem Cloud Data Migration Tool wurde ein Online Katalog für Cloud Data Hosting Solutions angelegt. Dieser beinhaltet ausführliche Beschreibungen und Kategorisierungen, beruhend auf Anbieterangaben und eigenen Erfahrungen mit den Dienstangeboten. Weiterhin unterstützt das Online Werkzeug bei der Cloud Datenmigration sowohl in Hinblick auf die Auswahl einer passenden Cloud Data Hosting Solution, als auch bei der Anpassung der Netzwerk-, Daten- und Anwendungsschicht und der Migration der Daten selbst.

7.2 Ausblick

Diese Arbeit betrachtet keine Herausforderungen bezüglich Datensicherheit während der Datenmigration. Diese werden in dem Konzept Cloud Data Protection as a Service [154] angesprochen und weiter entwickelt.

Um die Anpassungen der Anwendungen im Zuge der Datenmigration möglichst unabhängig vom verwendeten Anbieter und Dienst zu machen, können Cloud Abstraktionsschichten wie Deltacloud [155] oder jclouds [154] Unterstützung bieten. Deltacloud unterstützt z.B. aktuell BLOB- und Laufwerksspeicher Amazon S3, Eucalyptus Walrus, Rackspace CloudFiles, Microsoft Azure und Google Storage.

Ein weiterer Ansatz ist das Cloud Data Management Interface (CDMI) der Storage Networking Industry Association (SNIA). Dieser standardisiert Tags und Metadaten mit denen man seine Daten beschreiben kann und damit dem Anbieter sagt, wie die Daten z.B. bezüglich Backup, Archivierung oder Verschlüsselung behandelt werden sollen. [155]

Eine sinnvolle Erweiterung der Cloud Datenmigrationsmethodik ist die gleichzeitige Auswahl mehrere Migrationsszenarien. Dabei gilt es Abhängigkeiten zwischen den einzelnen Migrationsszenarien zu ermitteln. Zusätzlich ist eine Betrachtung mehrerer Quell- und Zielsystem innerhalb einer Datenmigration sinnvoll, um z.B. die Daten aus einer relationalen Datenbank ja nach Anwendungsfall aufzuteilen in einen BLOB, NoSQL und relationalen Cloud Datenspeicher.

Das Cloud Data Migration Tool kann auch erweitert werden um z.B. die Anforderungen an eine Cloud Data Hosting Solution, wie auch die Beschreibungen von Cloud Data Stores nach WS-Policy zu exportieren. Mit dieser Beschreibung könnte ein Enterprise Service Bus (ESB) dann automatisch einen Cloud Data Store auswählen oder eine Liste

an passenden Cloud Data Stores zurückgeben um Daten gemäß den Anforderungen in der Cloud zu speichern.

Des Weiteren kann das Cloud Datenmigrations-Tool um Adaptoren erweitert werden, welche mehr Cloud Data Stores anbinden. Ebenfalls ist eine detaillierte Beschreibung mit Hintergrundinformationen des ganzheitlichen Migrationsprozesses, der Szenarien, Cloud Data Hosting Solution Eigenschaften, Cloud Data Patterns und Auswirkungen auf höhere Anwendungsschichten, in Form eines Wikis, ergänzend zu den Fragebögen, sinnvoll. Für die Auswahl des Cloud Data Stores ist eine bessere Übersicht und Gegenüberstellung der in Frage kommenden Cloud Data Stores notwendig. Auch die Definition von KO-Kriterien oder eine Priorisierung der Cloud Data Hosting Solution Kriterien wäre hier hilfreich. Für eine höhere Relevanz bei den Vorschlägen zur Anpassung der Netzwerk-, Daten- und Anwendungsschicht, sollte nicht nur das ausgewählte Migrationsszenario, sondern auch die detailliertere Migrationsstrategie, der ausgewählte Cloud Data Store, sowie die Beschreibung des lokalen DBL in Betracht gezogen werden.

Die Ausführung der eigentlichen Datenmigration sollte ähnlich wie bei Informatica Cloud [14] nicht in der Cloud ausgeführt werden, sondern mit Hilfe eines lokal ausführbaren Migrationsagenten. Dies bringt sowohl Sicherheitsvorteile, da sensible Daten nicht über einen dritten Anbieter geschleust werden als auch Geschwindigkeitsvorteile, da die Daten direkt vom Quell- zum Zielspeicher transportiert werden können.

8 Anhang

8.1 Anhang 1

Project: Cloud Data Migration Tool (Data Usage from the Cloud) (IAAS)

Description: The Cloud Data Migration Tool (CDM) allows for migrating the data layer of an application to a cloud data store. The application itself is a Java Web application which uses a local MySQL database as data layer. The migration of the whole application to the Cloud should allow to use a compatible data store in the cloud.

Step 1a: Select Migration Scenario
Scenario: Data Usage from the Cloud

Step 1b: Refine Cloud Data Migration Strategy
Migration Strategy: Non-Live (with downtime)
Local DBL: Moved
Migration Degree: Complete
Source Data Store Type: RDBMS
Target Data Store Type: RDBMS
Product/Version Change: Same product, different version
Direction of Data Movement: From local datacenter to cloud
Migration Durability: Permanent
Migration Duration: Cut-off date
Resource Utilization: Acyclic variable
Utilization Increase: Slow

Step 1c: Identify Potential Migration Strategy Conflicts
Great, there seem to be no conflicts in your Cloud data migration strategy.

Step 2: Describe Desired Cloud Data Hosting Solution
Scalability
Degree of Automation: Partial Automation (Mix of Aut. and Manual)

Availability
Replication: Yes (distributed infrastructure)
Replication Method: Synchronous
Replication Location: Different Data Center in Different Region
Automatic Failover: Yes
Degree: 99.9%

Security
Storage Encryption: Yes
Transfer Encryption: Yes
Firewall: Yes
Authentication: Yes
Confidentiality: Yes
Integrity: Yes

Location
Cloud Location: Off-Premise

Interoperability
Data Portability: Import and Export
Storage Access: SQL
ORM: JPA, JDO
Migration & Deployment Support: Yes
Developer SDKs: Java, None Needed (Standard Support)

Compatibility
Product including Version: MySQL

Storage
Storage Type: RDBMS
Transaction Support: ACID

CAP (consistency, availability, partition tolerance)
Consistency Model: Strong
Availability in case of Partitioning: Available

Flexibility
Schema: Yes
Schema Customizable: Yes

Cloud Computing
Deployment Model: Private Cloud, Public Cloud, Community Cloud, Hybrid Cloud

Management and Maintenance effort
Degree of Automation: Managed/Automated

Backup
Backup Interval: Periodic, every x minutes: 1440
Number of Backups Kept: Number of backups: 30, Backups for number of days: 30
Backup Method: Snapshot (file system)

Step 3: Select Cloud Data Store
Name: Google Cloud SQL
Provider: Google

Step 4a: Describe Local Data Layer
Characterization
Location of the source system: On-premise
Cloud computing deployment model of the source system: None (not hosted in the cloud)
Cloud computing service model of the source system: None (not hosted in the cloud)
Does the source system run on a distributed infrastructure? No
Does the source system have a synchronization function? No
What is the data store type of the source system? RDBMS
Is the source system addressed on the network level via an URL that can be resolved by a DNS? No

RDBMS
Does the source system use proprietary extensions to the SQL 1999 standard and are these extensions used by the application? No
Does the source system support stored procedures or trigger and are these used by the application? No
Does the source system support transactions and are these used by the application? No
Does the source system support Joins and are there used by the application? No

Step 4b: Identify Patterns to Solve Potential Migration Conflicts
Conflict #1
You selected 'Acyclic variable' for the cloud data migration criterion 'Resource Utilization' and your cloud data store has 'Limited' selected for the criterion 'Degree'.
Possible Solution #1
Adaption of application logic. If your migration scenario requires scalability of your target system, but it isn't, you can adapt your application logic to switch between read and write replicas.
Conflict #2
Your cloud data store has 'No' selected for the criterion 'Confidentiality'.
Possible Solution #2
Adaption of the data access layer. If your target system does not ensure confidentiality, you can adapt your data access layer to prohibit storing sensitive data there.
Conflict #3
Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #3
Pseudonymizer of Critical Data. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Pseudonymizer of Critical Data' to ensure your critical data is not exposed.
Conflict #4
Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #4
Anonymizer of Critical Data. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Anonymizer of Critical Data' to ensure your critical data is not exposed.
Conflict #5
Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #5
Adaption of application logic. If your target system is hosted in a public cloud, you can adapt your application logic to prohibit storing sensitive data there.
Conflict #6
Your cloud data store has 'No' selected for the criterion 'Confidentiality'.
Possible Solution #6
Pseudonymizer of Critical Data. If your target system does not ensure confidentiality, you can use a cloud data pattern called 'Pseudonymizer of Critical Data'.
Conflict #7
Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #7
Filter of Critical Data. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Filter of Critical Data' to ensure your critical data is not exposed.
Conflict #8

Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #8
Confidentiality Level Data Aggregator. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Confidentiality Level Data Aggregator' to ensure your critical data is not exposed.
Conflict #9
Your cloud data store has 'No' selected for the criterion 'Confidentiality'.
Possible Solution #9
Anonymizer of Critical Data. If your target system does not ensure confidentiality, you can use a cloud data pattern called 'Anonymizer of Critical Data'.
Conflict #10
Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #10
Adaption of the data access layer. If your target system is hosted in a public cloud, you can adapt your data access layer to prohibit storing sensitive data there.
Conflict #11
Your cloud data store has 'No' selected for the criterion 'Confidentiality'.
Possible Solution #11
Adaption of application logic. If your target system does not ensure confidentiality, you can adapt your application layer to prohibit storing sensitive data there.
Conflict #12
You selected 'Acyclic variable' for the cloud data migration criterion 'Resource Utilization' and your cloud data store has 'Limited' selected for the criterion 'Degree'.
Possible Solution #12
Local Sharding-Based Router. If your migration scenario requires sharding functionality in your target system, but it isn't supported, you can use a cloud data pattern called "Local Sharding-Based Router".
Conflict #13
You selected 'Product/Version Change' for the cloud data migration criterion 'Same product, different version'.
Possible Solution #13
Adaption of application logic. If your current system is different (version) than your target system, then you can adapt your application logic to not use features that are not supported and use different semantics as required by the target system.
Conflict #14
Your cloud data store has 'No' selected for the criterion 'Confidentiality'.
Possible Solution #14
Confidentiality Level Data Aggregator. If your target system does not ensure confidentiality, you can use a cloud data pattern called 'Confidentiality Level Data Aggregator'.
Conflict #15
You selected 'Acyclic variable' for the cloud data migration criterion 'Resource Utilization' and your cloud data store has 'Limited' selected for the criterion 'Degree'.
Possible Solution #15
Local Database Proxy. If your migration scenario requires scalability of your target system, but it isn't, you can use a cloud data pattern called "Local Database Proyx".
Conflict #16
You selected 'Product/Version Change' for the cloud data migration criterion 'Same product, different version'.
Possible Solution #16

Adaption of the data access layer. If your current system is different (version) than your target system, then you can adapt your data access layer to simulate the missing features and translate the different semantics.
Conflict #17
You selected 'No' for the local data base criterion 'Is the source system addressed on the network level via an URL that can be resolved by a DNS?'.
Possible Solution #17
Change connection strings. If your source and target system is addressed via IP and the source and target data store are compatible you can adapt the connection strings in the application settings and configure your firewall to allow access to the target data store.
Conflict #18
You selected 'Acyclic variable' for the cloud data migration criterion 'Resource Utilization' and your cloud data store has 'Limited' selected for the criterion 'Degree'.
Possible Solution #18
Adaption of application logic. If your migration scenario requires sharding functionality in your target system, but it isn't supported, you can adapt your application logic to select the right shard for every query.
Conflict #19
Your cloud data store has 'No' selected for the criterion 'Confidentiality'.
Possible Solution #19
Filter of Critical Data. If your target system does not ensure confidentiality, you can use a cloud data pattern called 'Filter of Critical Data'.
Conflict #20
You selected 'Acyclic variable' for the cloud data migration criterion 'Resource Utilization' and your cloud data store has 'Limited' selected for the criterion 'Degree'.
Possible Solution #20
Adaption of the data access layer. If your migration scenario requires scalability of your target system, but it isn't, you can adapt your data access layer to switch between read and write replicas.
Conflict #21
You selected 'Product/Version Change' for the cloud data migration criterion 'Same product, different version'.
Possible Solution #21
Data Store Functionality Extension. If your current system is different (version) than your target system, then a cloud data pattern called "Data Store Functionality Extension" can simulate features and semantics that the target system does not understand or would handle differently.
Conflict #22
You selected 'Acyclic variable' for the cloud data migration criterion 'Resource Utilization' and your cloud data store has 'Limited' selected for the criterion 'Degree'.
Possible Solution #22
Adaption of the data access layer. If your migration scenario requires sharding functionality in your target system, but it isn't supported, you can adapt your data access layer to select the right shard for every query.

Step 5: Adapt Data Access Layer And Upper Application Layers
Data Usage from the Cloud - Network Layer
CNAME

If the source and target system are compatible and the source system is currently addressed via an URL, a change of the CNAME entry to point to the new target system is sufficient to allow all applications to use the new target system.
DNS Cache TTL
If the CNAME entry can be adapted to point to the new target system, make sure the DNS cache of all applications that use the old source system have a short time to live (TTL).
Firewall (Open Ports)
If the new target system is available under a different IP address or port then firewall settings need to be updated to allow all applications to access the new target system. It can also be useful to setup a Virtual Private Cloud (VPC). Make sure that not only the firewall settings for applications that use the old source system are adapted, but also the firewall settings for failover configurations, monitoring scripts and backup scripts.

Data Usage from the Cloud - Data Access Layer
Connection Strings
If the source and target system are compatible it can be sufficient to change the connection strings of all applications using the old source system to point to the address of the new target system. Make sure that not only applications that use the old source system are adapted, but also failover configurations, monitoring scripts and backup scripts.
Cloud Data Pattern
If the source and target system are not compatible, Cloud Data Patterns can be used to simulate missing functionality, e.g. non-standard SQL commands.
Synchronisation
In the geographic replication scenario the data access layer and above layers like the application layer have to be aware that the same layers run in parallel at different Cloud data centers and that the database layer has to be synchronized in the background.
In the data synchronisation scenario the synchronisation between the local replica and the Cloud master database has to be setup. This can be done by a synchronisation functionality of the RDBMS or by the data access layer or an external synchronisation tool.
Semantic (Schema vs. DB Name)
Some data stores have a different semantic what a database and schema is (e.g. Oracle and SQL Server). This can be translated by the data access layer without adapting any upper application layers.
Naming Conventions
Most data stores have different naming conventions, therefore short, uppercase identifiers (table names, row names, constraint names) without special characters and potentially reserved words should be used. The data access layer can transform denied identifiers on the fly to allowed ones without the need to adapt upper application layers.
Data Types
Most data stores support different sets of data types which the data access layer can transform on the fly without the need to change upper application layers (e.g. BOOLEAN to BIT or CHAR(1)).
Semantics ("" vs. Null)
Some databases don't support null values for data entries, if the previous data store supported these, the data access layer can in some cases resolve that conflict if the applications itself don't differentiate between 'null' and empty values. If the applications differentiate between 'null' and empty values, the upper application layers have to be adopted.
Sorting

If the target system is a NoSQL datastore and supports sorting of result sets, it may be necessary to add zero-padding for numbers and an offset to columns with negative numbers if the ordering is done in lexicographical order.
Attribute Size
If the target system is a NoSQL datastore and allows only small values for attributes (e.g. max. 1kB), then larger values have to be stored in a Blob store and a pointer to the location in the Blob store can work as an intermediary attribute.

Data Usage from the Cloud - Application Logic Layer
Data Store Type Change
If the type of the data store (RDBMS, NoSQL, Blob) changes during the data migration it is usually not sufficient enough to simulate missing functionality in the data access layer. If for example the schema is no fixed, or joins, strict consistency and ACID transactions are not supported any more, the application logic has to be adapted to reflect the conceptual changes. In case of migrating from a RDBMS to a NoSQL and Blob store, a small part of the data could still be required to be stored in a RDBMS to support e.g. financial transactions.
Cloud Data Pattern Effects
If Cloud Data Patterns for Confidentiality are used in the data access layer, the application layer has to be aware that it might not have a full view on the actual data, since it could be anonymized, filtered or pseudonymized.
Product Change (Data Types, etc.)
If the product of the data store changes during the migration the semantics of e.g. comparison operators could have changed which cannot be translated by the data access layer and requires the application layer to be adopted.
Semantics (Comparisions, Locks)
Despite SQL being a standardised language, most vendors add product specific SQL commands or differ in the semantics of e.g. locks to allow dirty reads or more fine grained locks. These differences have to be identified during the migration and applied to the application layer.

Step 6: Migrate Data to the Selected Cloud Data Store
Source System
MYSQL_LOCAL_SOURCE
Username: root
Password: password
Host: localhost
Port: 3306
Database: clouddatamigration
Compatibility_Mode: -

Target System
GOOGLE_CLOUD_SQL_TARGET

8.2 Anhang 2

Step 1b: Refine Cloud Data Migration Strategy
Product/Version Change: Same product, same version

Step 4b: Identify Patterns to Solve Potential Migration Conflicts

Conflict #1: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #1: Pseudonymizer of Critical Data. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Pseudonymizer of Critical Data' to ensure your critical data is not exposed.
Conflict #2: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #2: Anonymizer of Critical Data. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Anonymizer of Critical Data' to ensure your critical data is not exposed.
Conflict #3: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #3: Adaption of application logic. If your target system is hosted in a public cloud, you can adapt your application logic to prohibit storing sensitive data there.
Conflict #4: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #4: Filter of Critical Data. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Filter of Critical Data' to ensure your critical data is not exposed.
Conflict #5: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #5: Confidentiality Level Data Aggregator. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Confidentiality Level Data Aggregator' to ensure your critical data is not exposed.
Conflict #6: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #6: Adaption of the data access layer. If your target system is hosted in a public cloud, you can adapt your data access layer to prohibit storing sensitive data there.
Conflict #7: You selected 'No' for the local data base criterion 'Is the source system addressed on the network level via an URL that can be resolved by a DNS?'.
Possible Solution #7: Change connection strings. If your source and target system is addressed via IP and the source and target data store are compatible you can adapt the connection strings in the application settings and configure your firewall to allow access to the target data store.

Step 6: Migrate Data to the Selected Cloud Data Store
Source System
VIRTUAL_MACHINE_IMAGE_SOURCE
In order to migrate an existing virtual machine image you need to follow the import instructions from the target system provider.

Target System
AWS_EC2_IMPORT_TARGET
In order to import an existing virtual machine image follow the instructions provided by Amazon Web Services. In a nutshell you need to download the 'VM Import Command Line Tools', import the VMDK, VHD or RAW file via the ec2-import-instance API, retrieve the Amazon EC2 instance id, launch the image from S3 (instance store-backed) and persist it's state to EBS and delete the image file from S3.

8.3 Anhang 3

Step 1b: Refine Cloud Data Migration Strategy
Product/Version Change: Different products

Step 2: Describe Desired Cloud Data Hosting Solution
Compatibility
Product including Version: MS SQL (SQL Server)

Step 4b: Identify Patterns to Solve Potential Migration Conflicts
Conflict #1: You selected 'Acyclic variable' for the cloud data migration criterion 'Resource Utilization' and your cloud data store has 'Limited' selected for the criterion 'Degree'.
Possible Solution #1: Adaption of application logic. If your migration scenario requires scalability of your target system, but it isn't, you can adapt your application logic to switch between read and write replicas.
Conflict #2: Your cloud data store has 'No' selected for the criterion 'Confidentiality'.
Possible Solution #2: Adaption of the data access layer. If your target system does not ensure confidentiality, you can adapt your data access layer to prohibit storing sensitive data there.
Conflict #3: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #3: Pseudonymizer of Critical Data. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Pseudonymizer of Critical Data' to ensure your critical data is not exposed.
Conflict #4: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #4: Anonymizer of Critical Data. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Anonymizer of Critical Data' to ensure your critical data is not exposed.
Conflict #5: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #5: Adaption of application logic. If your target system is hosted in a public cloud, you can adapt your application logic to prohibit storing sensitive data there.
Conflict #6: Your cloud data store has 'No' selected for the criterion 'Confidentiality'.
Possible Solution #6: Pseudonymizer of Critical Data. If your target system does not ensure confidentiality, you can use a cloud data pattern called 'Pseudonymizer of Critical Data'.
Conflict #7: You selected 'Product/Version Change' for the cloud data migration criterion 'Different products'.
Possible Solution #7: Data Store Functionality Extension. If your current system is different (product) than your target system, then a cloud data pattern called "Data Store Functionality Extension" can simulate features and semantics that the target system does not understand or would handle differently.
Conflict #8: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #8: Filter of Critical Data. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Filter of Critical Data' to ensure your critical data is not exposed.

Conflict #9: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #9: Confidentiality Level Data Aggregator. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Confidentiality Level Data Aggregator' to ensure your critical data is not exposed.
Conflict #10: Your cloud data store has 'No' selected for the criterion 'Confidentiality'.
Possible Solution #10: Anonymizer of Critical Data. If your target system does not ensure confidentiality, you can use a cloud data pattern called 'Anonymizer of Critical Data'.
Conflict #11: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #11: Adaption of the data access layer. If your target system is hosted in a public cloud, you can adapt your data access layer to prohibit storing sensitive data there.
Conflict #12: Your cloud data store has 'No' selected for the criterion 'Confidentiality'.
Possible Solution #12: Adaption of application logic. If your target system does not ensure confidentiality, you can adapt your application layer to prohibit storing sensitive data there.
Conflict #13: You selected 'Acyclic variable' for the cloud data migration criterion 'Resource Utilization' and your cloud data store has 'Limited' selected for the criterion 'Degree'.
Possible Solution #13: Local Sharding-Based Router. If your migration scenario requires sharding functionality in your target system, but it isn't supported, you can use a cloud data pattern called "Local Sharding-Based Router".
Conflict #14: Your cloud data store has 'No' selected for the criterion 'Confidentiality'.
Possible Solution #14: Confidentiality Level Data Aggregator. If your target system does not ensure confidentiality, you can use a cloud data pattern called 'Confidentiality Level Data Aggregator'.
Conflict #15: You selected 'Acyclic variable' for the cloud data migration criterion 'Resource Utilization' and your cloud data store has 'Limited' selected for the criterion 'Degree'.
Possible Solution #15: Local Database Proxy. If your migration scenario requires scalability of your target system, but it isn't, you can use a cloud data pattern called "Local Database Proxy".
Conflict #16: You selected 'No' for the local data base criterion 'Is the source system addressed on the network level via an URL that can be resolved by a DNS?'.
Possible Solution #16: Change connection strings. If your source and target system is addressed via IP and the source and target data store are compatible you can adapt the connection strings in the application settings and configure your firewall to allow access to the target data store.
Conflict #17: You selected 'Acyclic variable' for the cloud data migration criterion 'Resource Utilization' and your cloud data store has 'Limited' selected for the criterion 'Degree'.
Possible Solution #17: Adaption of application logic. If your migration scenario requires sharding functionality in your target system, but it isn't supported, you can adapt your application logic to select the right shard for every query.
Conflict #18: Your cloud data store has 'No' selected for the criterion 'Confidentiality'.
Possible Solution #18: Filter of Critical Data. If your target system does not ensure confidentiality, you can use a cloud data pattern called 'Filter of Critical Data'.

Conflict #19: You selected 'Acyclic variable' for the cloud data migration criterion 'Resource Utilization' and your cloud data store has 'Limited' selected for the criterion 'Degree'.
Possible Solution #19: Adaption of the data access layer. If your migration scenario requires scalability of your target system, but it isn't, you can adapt your data access layer to switch between read and write replicas.
Conflict #20: You selected 'Product/Version Change' for the cloud data migration criterion 'Different products'.
Possible Solution #20: Adaption of the data access layer. If your current system is different (product) than your target system, then you can adapt your data access layer to simulate the missing features and translate the different semantics.
Conflict #21: You selected 'Acyclic variable' for the cloud data migration criterion 'Resource Utilization' and your cloud data store has 'Limited' selected for the criterion 'Degree'.
Possible Solution #21: Adaption of the data access layer. If your migration scenario requires sharding functionality in your target system, but it isn't supported, you can adapt your data access layer to select the right shard for every query.
Conflict #22: You selected 'Product/Version Change' for the cloud data migration criterion 'Different products'.
Possible Solution #22: Adaption of application logic. If your current system is different (product) than your target system, then you can adapt your application logic to not use features that are not supported and use different semantics as required by the target system.

Step 6: Migrate Data to the Selected Cloud Data Store
Source System
MYSQL_LOCAL_SOURCE
Username: root
Password: password
Host: localhost
Port: 3306
Database: clouddatamigration
Compatibility_Mode: -

Target System
AZURE_SQL_DATABASE_SERVICE_TARGET
Username: *username*
Password: *password*
Host: *servername*.database.azure.com
Port: 1433
Database: clouddatamigration

8.4 Anhang 4

Step 1a: Select Migration Scenario
Scenario: Plain Outsourcing

Step 1b: Refine Cloud Data Migration Strategy
Product/Version Change: Same product, same version

Step 2: Describe Desired Cloud Data Hosting Solution

Cloud Computing
Service Model: IaaS

Step 3: Select Cloud Data Store
Name: Amazon EC2
Provider: Amazon Web Services (AWS)

Step 4b: Identify Patterns to Solve Potential Migration Conflicts
Conflict #1: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #1: Pseudonymizer of Critical Data. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Pseudonymizer of Critical Data' to ensure your critical data is not exposed.
Conflict #2: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #2: Anonymizer of Critical Data. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Anonymizer of Critical Data' to ensure your critical data is not exposed.
Conflict #3: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #3: Adaption of application logic. If your target system is hosted in a public cloud, you can adapt your application logic to prohibit storing sensitive data there.
Conflict #4: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #4: Filter of Critical Data. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Filter of Critical Data' to ensure your critical data is not exposed.
Conflict #5: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #5: Confidentiality Level Data Aggregator. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Confidentiality Level Data Aggregator' to ensure your critical data is not exposed.
Conflict #6: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #6: Adaption of the data access layer. If your target system is hosted in a public cloud, you can adapt your data access layer to prohibit storing sensitive data there.
Conflict #7: You selected 'No' for the local data base criterion 'Is the source system addressed on the network level via an URL that can be resolved by a DNS?'.
Possible Solution #7: Change connection strings. If your source and target system is addressed via IP and the source and target data store are compatible you can adapt the connection strings in the application settings and configure your firewall to allow access to the target data store.

Step 6: Migrate Data to the Selected Cloud Data Store
Source System
VIRTUAL_MACHINE_IMAGE_SOURCE
In order to migrate an existing virtual machine image you need to follow the import instructions from the target system provider.

Target System
AWS_EC2_IMPORT_TARGET
In order to import an existing virtual machine image follow the instructions provided by Amazon Web Services. In a nutshell you need to download the 'VM Import Command Line Tools', import the VMDK, VHD or RAW file via the ec2-import-instance API, retrieve the Amazon EC2 instance id, launch the image from S3 (instance store-backed) and persist it's state to EBS and delete the image file from S3.

8.5 Anhang 5

Step 1a: Select Migration Scenario
Scenario: Usage of highly-scalable data stores (NoSQL, Blob Store)

Step 1b: Refine Cloud Data Migration Strategy
Product/Version Change: Different products
Target Data Store Type: NoSQL

Step 2: Describe Desired Cloud Data Hosting Solution
Scalability
Degree: Virtually Unlimited
Interoperability
Data Exchange Format: XML
Storage Access: SOA
Storage
Storage Type: NoSQL
Performance
Predictability Read/Write/Response: Predictable, x milliseconds: 10
CAP (consistency, availability, partition tolerance)
Consistency Model: Eventual Consistency

Step 3: Select Cloud Data Store
Name: Amazon SimpleDB
Provider: Amazon Web Services (AWS)

Step 4b: Identify Patterns to Solve Potential Migration Conflicts
Conflict #1: You selected 'Acyclic variable' for the cloud data migration criterion 'Resource Utilization' and your cloud data store has 'Limited' selected for the criterion 'Degree'.
Possible Solution #1: Adaption of application logic. If your migration scenario requires scalability of your target system, but it isn't, you can adapt your application logic to switch between read and write replicas.
Conflict #2: Your cloud data store has 'No' selected for the criterion 'Confidentiality'.
Possible Solution #2: Adaption of the data access layer. If your target system does not ensure confidentiality, you can adapt your data access layer to prohibit storing sensitive data there.
Conflict #3: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #3: Pseudonymizer of Critical Data. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Pseudonymizer of Critical Data' to ensure your critical data is not exposed.

Conflict #4: You selected 'RDBMS' for the cloud data migration criterion 'Source Data Store Type' and 'NoSQL' for the cloud data migration criterion 'Target Data Store Type'.
Possible Solution #4: Adaption of application logic. If your current system is a RDBMS and uses stored procedures or triggers and your target system is a NoSQL data store that does not support stored procedures or triggers, you can adapt your application logic to process the stored procedure and trigger logic.
Conflict #5: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #5: Anonymizer of Critical Data. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Anonymizer of Critical Data' to ensure your critical data is not exposed.
Conflict #6: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #6: Adaption of application logic. If your target system is hosted in a public cloud, you can adapt your application logic to prohibit storing sensitive data there.
Conflict #7: Your cloud data store has 'No' selected for the criterion 'Confidentiality'.
Possible Solution #7: Pseudonymizer of Critical Data. If your target system does not ensure confidentiality, you can use a cloud data pattern called 'Pseudonymizer of Critical Data'.
Conflict #8: You selected 'RDBMS' for the local data base criterion 'What is the data store type of the source system?' and your cloud data store has 'NoSQL' selected for the criterion 'Storage Type'.
Possible Solution #8: Adaption of application logic. If your source system is a RDBMS and target system a NoSQL data store and you use complex SQL queries which are not supported in the NoSQL store, you can adapt your application logic to use simpler request and applying the query logic on the client side.
Conflict #9: You selected 'Product/Version Change' for the cloud data migration criterion 'Different products'.
Possible Solution #9: Data Store Functionality Extension. If your current system is different (product) than your target system, then a cloud data pattern called "Data Store Functionality Extension" can simulate features and semantics that the target system does not understand or would handle differently.
Conflict #10: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #10: Filter of Critical Data. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Filter of Critical Data' to ensure your critical data is not exposed.
Conflict #11: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.
Possible Solution #11: Confidentiality Level Data Aggregator. If your target system is hosted in a public cloud, you can use a cloud data pattern called 'Confidentiality Level Data Aggregator' to ensure your critical data is not exposed.
Conflict #12: Your cloud data store has 'No' selected for the criterion 'Confidentiality'.
Possible Solution #12: Anonymizer of Critical Data. If your target system does not ensure confidentiality, you can use a cloud data pattern called 'Anonymizer of Critical Data'.
Conflict #13: Your cloud data store has 'Public Cloud' selected for the criterion 'Deployment Model'.

Possible Solution #13: Adaption of the data access layer. If your target system is hosted in a public cloud, you can adapt your data access layer to prohibit storing sensitive data there.
Conflict #14: Your cloud data store has 'No' selected for the criterion 'Confidentiality'.
Possible Solution #14: Adaption of application logic. If your target system does not ensure confidentiality, you can adapt your application layer to prohibit storing sensitive data there.
Conflict #15: You selected 'Acyclic variable' for the cloud data migration criterion 'Resource Utilization' and your cloud data store has 'Limited' selected for the criterion 'Degree'.
Possible Solution #15: Local Sharding-Based Router. If your migration scenario requires sharding functionality in your target system, but it isn't supported, you can use a cloud data pattern called "Local Sharding-Based Router".
Conflict #16: Your cloud data store has 'No' selected for the criterion 'Confidentiality'.
Possible Solution #16: Confidentiality Level Data Aggregator. If your target system does not ensure confidentiality, you can use a cloud data pattern called 'Confidentiality Level Data Aggregator'.
Conflict #17: You selected 'Acyclic variable' for the cloud data migration criterion 'Resource Utilization' and your cloud data store has 'Limited' selected for the criterion 'Degree'.
Possible Solution #17: Local Database Proxy. If your migration scenario requires scalability of your target system, but it isn't, you can use a cloud data pattern called "Local Database Proxy".
Conflict #18: You selected 'RDBMS' for the cloud data migration criterion 'Source Data Store Type' and 'NoSQL' for the cloud data migration criterion 'Target Data Store Type'.
Possible Solution #18: Adaption of the data access layer. If your current system is a RDBMS and uses stored procedures or triggers and your target system is a NoSQL data store that does not support stored procedures or triggers, you can adapt your data access layer to process the stored procedure and trigger logic.
Conflict #19: You selected 'No' for the local data base criterion 'Is the source system addressed on the network level via an URL that can be resolved by a DNS?'.
Possible Solution #19: Change connection strings. If your source and target system is addressed via IP and the source and target data store are compatible you can adapt the connection strings in the application settings and configure your firewall to allow access to the target data store.
Conflict #20: You selected 'Acyclic variable' for the cloud data migration criterion 'Resource Utilization' and your cloud data store has 'Limited' selected for the criterion 'Degree'.
Possible Solution #20: Adaption of application logic. If your migration scenario requires sharding functionality in your target system, but it isn't supported, you can adapt your application logic to select the right shard for every query.
Conflict #21: Your cloud data store has 'No' selected for the criterion 'Confidentiality'.
Possible Solution #21: Filter of Critical Data. If your target system does not ensure confidentiality, you can use a cloud data pattern called 'Filter of Critical Data'.
Conflict #22: You selected 'Acyclic variable' for the cloud data migration criterion 'Resource Utilization' and your cloud data store has 'Limited' selected for the criterion 'Degree'.
Possible Solution #22: Adaption of the data access layer. If your migration scenario requires scalability of your target system, but it isn't, you can adapt your data access layer to switch between read and write replicas.

Conflict #23: You selected 'Product/Version Change' for the cloud data migration criterion 'Different products'.
Possible Solution #23: Adaption of the data access layer. If your current system is different (product) than your target system, then you can adapt your data access layer to simulate the missing features and translate the different semantics.
Conflict #24: You selected 'RDBMS' for the cloud data migration criterion 'Source Data Store Type' and 'NoSQL' for the cloud data migration criterion 'Target Data Store Type'.
Possible Solution #24: Emulator of Stored Procedures. If your current system is a RDBMS and uses stored procedures or triggers and your target system is a NoSQL data store that does not support stored procedures or triggers, you can use a cloud data pattern called "Emulator of Stored Procedures" that interprets stored procedures and triggers.
Conflict #25: You selected 'RDBMS' for the local data base criterion 'What is the data store type of the source system?' and your cloud data store has 'NoSQL' selected for the criterion 'Storage Type'.
Possible Solution #25: External services for complex queries. If your source system is a RDBMS and target system a NoSQL data store and you use complex SQL queries which are not supported in the NoSQL store, you can use external services that allow complex queries on NoSQL databases like AWS Elastic MapReduce.
Conflict #26: You selected 'RDBMS' for the local data base criterion 'What is the data store type of the source system?' and your cloud data store has 'NoSQL' selected for the criterion 'Storage Type'.
Possible Solution #26: Adaption of the data access layer. If your source system is a RDBMS and target system a NoSQL data store and you use complex SQL queries which are not supported in the NoSQL store, you can adapt your data access layer to simulate those complex queries by splitting them into simpler request and applying the query logic on the client side.
Conflict #27: You selected 'Acyclic variable' for the cloud data migration criterion 'Resource Utilization' and your cloud data store has 'Limited' selected for the criterion 'Degree'.
Possible Solution #27: Adaption of the data access layer. If your migration scenario requires sharding functionality in your target system, but it isn't supported, you can adapt your data access layer to select the right shard for every query.
Conflict #28: You selected 'Product/Version Change' for the cloud data migration criterion 'Different products'.
Possible Solution #28: Adaption of application logic. If your current system is different (product) than your target system, then you can adapt your application logic to not use features that are not supported and use different semantics as required by the target system.

9 Literaturverzeichnis

[1] R. Mietzner, T. Unger, R. Titze, and F. Leymann, "Combining Different Multi-Tenancy Patterns in Service-Oriented Applications," *Proceedings of the 13th IEEE Enterprise Distributed Object Conference (EDOC 2009)*, pp. 131–140, Sep. 2009.

[2] C. J. Guo, W. Sun, Y. Huang, Z. H. Wang, and B. Gao, "A Framework for Native Multi-Tenancy Application Development and Management," *E-Commerce Technology, IEEE International Conference on, and Enterprise Computing, E-Commerce, and E-Services, IEEE International Conference on*, vol. 0, pp. 551–558, 2007.

[3] N. Leavitt, "Is Cloud Computing Really Ready for Prime Time?," *Computer*, vol. 42, no. 1, pp. 15–20, Jan. 2009.

[4] S. Strauch, U. Breitenbuecher, O. Kopp, F. Leymann, and T. Unger, "CLOUD DATA PATTERNS FOR CONFIDENTIALITY CLOUD DATA PATTERNS FOR CONFIDENTIALITY," *Architecture*, 2012.

[5] J. Dunkel, A. Eberhart, S. Fischer, C. Kleiner, and A. Koschel, *Systemarchitekturen für Verteilte Anwendungen. Client-Server, Multi-Tier, SOA, Event Driven Architectures, P2P, Grid, Web 2.0*. 2008, p. 42.

[6] B. Schwan, "Voll homomorphe Verschlüsselung in der Cloud."

[7] M. Fowler and P. Sadalage, *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Pearson Education, Limited, 2012.

[8] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services.," *ACM SIGACT News*, vol. 33, no. 2, pp. 51–59, 2002.

[9] J. Varia, "Migrating your Existing Applications to the AWS Cloud. A Phase-driven Approach to Cloud Migration," 2010.

[10] P. Mell and T. Grance, "The NIST Definition of Cloud Computing (Draft) Recommendations of the National Institute of Standards and Technology," *Nist Special Publication*, vol. 145.

[11] S. Strauch, O. Kopp, F. Leymann, and T. Unger, "A Taxonomy for Cloud Data Hosting Solutions," *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, pp. 577–584, Dec. 2011.

[12] S. Strauch, V. Andrikopoulos, U. Breitenbuecher, O. Kopp, and F. Leymann, "Data Layer Patterns for Cloud Applications (under review)," *Business*.

[13] M. Menzel, "CloudGenius : Decision Support for Web Server Cloud Migration Categories and Subject Descriptors," no. Vm, pp. 979–988, 2012.

[14] "Cloud Computing, Integration on Demand, Saas Data Integration Services - Informatica Cloud." [Online]. Available: http://www.informaticacloud.com/.

[15] “Apex Data Loader - developer.force.com.” [Online]. Available: http://wiki.developerforce.com/page/Apex_Data_Loader.

[16] D. S. Linthicum, “Approaching SaaS Integration with Data Integration Best Practices and Technology,” 2009.

[17] T. Laszewski and P. Nauduri, *Migrating to the Cloud: Oracle Client / Server Modernization*. Elsevier Science, 2011.

[18] T. Höllwarth, *Cloud Migration*. mitp/bhv, 2011, pp. 129–131.

[19] Netspective Communications LLC and Mercury Consulting, “Securing Your Cloud-Based Data Integration – A Best Practices Checklist,” pp. 1–10, 2011.

[20] Amazon Web Services, “Amazon RDS Customer Data Import Guide for MySQL: Articles & Tutorials: Amazon Web Services.” [Online]. Available: http://aws.amazon.com/articles/2933?_encoding=UTF8&jiveRedirect=1.

[21] DaveL, “AWS Developer Forums: Announcing Amazon DynamoDB,” 2012. [Online]. Available: https://forums.aws.amazon.com/ann.jspa?annID=1326.

[22] Amazon Web Services, “Amazon Glacier,” 2012. .

[23] Amazon Web Services, “Migration Scenario: Migrating Web Applications to the AWS Cloud.” [Online]. Available: http://media.amazonwebservices.com/CloudMigration-scenario-wep-app.pdf.

[24] Amazon Web Services, “Migration Scenario: Migrating Batch Processes to the AWS Cloud.” [Online]. Available: http://media.amazonwebservices.com/CloudMigration-scenario-batch-apps.pdf.

[25] Amazon Web Services, “Migration Scenario: Migrating Backend Processing Pipeline to the AWS Cloud.” [Online]. Available: http://media.amazonwebservices.com/CloudMigration-scenario-backend-processing.pdf.

[26] Microsoft Azure, “Parallel.” [Online]. Available: http://www.windowsazure.com/de-de/home/scenarios/parallel/.

[27] Microsoft Azure, “Verbraucher.” [Online]. Available: http://www.windowsazure.com/de-de/home/scenarios/consumer/.

[28] Microsoft Azure, “Enterprise – Szenarien.” [Online]. Available: http://www.windowsazure.com/de-de/home/scenarios/enterprise/.

[29] Microsoft Azure, “Mobil.” [Online]. Available: http://www.windowsazure.com/de-de/home/scenarios/mobile/.

[30] Microsoft Azure, “Spiele.” [Online]. Available: http://www.windowsazure.com/de-de/home/scenarios/gaming/.

[31] Microsoft Azure, “Big Data.” [Online]. Available: http://www.windowsazure.com/de-de/home/scenarios/big-data/.

[32] Microsoft Azure, “SaaS - Szenarien.” [Online]. Available: http://www.windowsazure.com/de-de/home/scenarios/saas/.

[33] Microsoft Azure, “Webszenarien.” [Online]. Available: http://www.windowsazure.com/de-de/home/scenarios/web/.

[34] S. R. Cunningham, “Windows Azure – Application Profile Guidance. Custom IIS Web / Microsoft SQL Server Application Migration Scenario,” 2010.

[35] S. R. Cunningham, “Windows Azure – Application Profile Guidance. Custom Web (Rapid Scaling Focus) Application Migration Scenario,” 2010.

[36] S. R. Cunningham, “Windows Azure – Application Profile Guidance. Custom E-Commerce (Elasticity Focus) Application Migration Scenario,” 2010.

[37] Microsoft Azure, “General Guidelines and Limitations (SQL Azure Database).” [Online]. Available: http://msdn.microsoft.com/en-us/library/windowsazure/ee336245.aspx.

[38] D. Nethi and M. Thomassy, “Developing and Deploying with SQL Azure,” 2010.

[39] “SQL Azure Migration Wizard v3.8.6.” [Online]. Available: http://sqlazuremw.codeplex.com/.

[40] “Move Access Data to the Cloud - TechNet Articles - Home - TechNet Wiki.” [Online]. Available: http://social.technet.microsoft.com/wiki/contents/articles/move-access-data-to-the-cloud.aspx.

[41] “SQL Azure Data Sync.” [Online]. Available: http://msdn.microsoft.com/en-us/library/hh456371.aspx.

[42] J. Lee, G. Malcolm, A. Matthews, and C. Master, “Overview of Microsoft SQL Azure Database,” 2009.

[43] T. Redkar and T. Guidici, *Windows Azure Platform*. Apress, 2011.

[44] M. Blain, “Bulkloader, aka Data Migration with App Engine,” 2010. [Online]. Available: http://dl.google.com/googleio/2010/app-engine-data-migration.pdf.

[45] “Hoch- und Herunterladen von Daten - Google App Engine - Google Code.” [Online]. Available: http://code.google.com/intl/de/appengine/docs/python/tools/uploadingdata.html.

[46] “Migrating Applications - Google App Engine - Google Code.” [Online]. Available: http://code.google.com/intl/de/appengine/docs/adminconsole/migration.html.

[47] "Auswählen eines Datenspeichers (Java) - Google App Engine - Google Code." [Online]. Available: http://code.google.com/intl/de/appengine/docs/java/datastore/hr/.

[48] "Backup/Restore, Copy, and Delete Data - Google App Engine - Google Code." [Online]. Available: http://code.google.com/intl/de/appengine/docs/adminconsole/datastoreadmin.html.

[49] "CRM, the cloud, and the social enterprise - Salesforce.com." [Online]. Available: http://www.salesforce.com/.

[50] "Learning Center - 5 steps to getting your data into Salesforce CRM - Salesforce.com." [Online]. Available: https://www.salesforce.com/customer-resources/learning-center/details/5-steps-getting-data.jsp.

[51] "CRC:Design Data Migration Source Systems and Tools - developer.force.com." [Online]. Available: http://wiki.developerforce.com/page/CRC:Design_Data_Migration_Source_Systems_and_Tools.

[52] "Salesforce Data Migration Workplan." [Online]. Available: http://www.developerforce.com/media/consultant/Data_Migration_projectWorkflows.jpg.

[53] "CRC:Design Data Migration Project Phases - developer.force.com." [Online]. Available: http://wiki.developerforce.com/page/CRC:Design_Data_Migration_Project_Phases.

[54] Beauftragte der Bundesregierung für Informationstechnik, "Migrationsleitfaden. Leitfaden für die Migration von Software," 2012.

[55] A. Pradesh, V. G. Reddy, and G. S. Kumar, "Cloud Computing with a Data Migration," *Journal of Current Computer Science and Technology*, vol. 1, no. 6, 2011.

[56] J. Morris, *Practical Data Migration*. British Computer Society, 2006.

[57] Germanviscuso and Yobot, "Versant Object Database - Wikipedia, the free encyclopedia." [Online]. Available: http://en.wikipedia.org/wiki/Versant_Object_Database#Data_distribution_architecture.

[58] V. T. K. Tran, K. Lee, A. Fekete, A. Liu, and J. Keung, "Size Estimation of Cloud Migration Projects with Cloud Migration Point (CMP)," *2011 International Symposium on Empirical Software Engineering and Measurement*, pp. 265–274, Sep. 2011.

[59] T. S. Mohan, "MIGRATING INTO A CLOUD," in *CLOUD COMPUTING. Principles and Paradigms*, 1st ed., R. Buyya, J. Broberg, and A. Goscinski, Eds. Hoboken, New Jersey: John Wiley & Sons, Inc., 2011.

[60] T. S. Mohan, “Migrating Scientific Applications from Grid and Cluster Computing into the Cloud. Issues & Challenges,” no. March, 2011.

[61] M. Hajjat, X. Sun, Y. Sung, D. Maltz, S. Rao, K. Sripandidkulchai, and M. Tawarmalani, “Cloudward Bound: Planning for Beneficial Migration of Enterprise Applications to the Cloud,” *ACM SIGCOMM*, 2010.

[62] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “Above the Clouds : A Berkeley View of Cloud Computing Cloud Computing : An Old Idea Whose Time Has (Finally) Come,” *Computing*, pp. 7–13, 2009.

[63] Netspective Communications LLC, “Informatica Cloud Architecture and Security Overview,” *Informatica*, pp. 1–12, 2011.

[64] “VISION Cloud - Fullstory.” [Online]. Available: http://visioncloud.eu/.

[65] VISOION Cloud, “VISOION Cloud. VIRTUALIZED STORAGE SERVICES FOUNDATION FOR THE FUTURE INTERNET. Deliverable D0.1.1b Project Progress Report,” pp. 1–82, 2011.

[66] D. S. Linthicum, “Replicating salesforce.com Data for Operational Reporting and Compliance,” 2010.

[67] P. Haettenschwiler, “Neues anwenderfreundliches Konzept der Entscheidungsunterstützung,” in *Gutes Entscheiden in Wirtschaft, Politik und Gesellschaft*, Zürich: vdf Hochschulverlag AG, 1999, pp. 189–208.

[68] D. J. Power, *Decision support systems: concepts and resources for managers*. Westport, Conn.: Quorum Books, 2002.

[69] A. Khajeh-Hosseini, I. Sommerville, J. Bogaerts, and P. Teregowda, “Decision Support Tools for Cloud Migration in the Enterprise,” *2011 IEEE 4th International Conference on Cloud Computing*, pp. 541–548, Jul. 2011.

[70] “cumulusgenius - CumulusGenius Cloud Migration Decision Support Tool.” [Online]. Available: http://code.google.com/p/cumulusgenius/.

[71] “aotearoadecisions - Aotearoa / The Multi-Goal Cloud Decision-Making Tool.” [Online]. Available: http://code.google.com/p/aotearoadecisions/.

[72] “Drools - JBoss Community.” [Online]. Available: http://www.jboss.org/drools.

[73] A. Farrag, R. Medlin, C. Moore, and J. Mostafa, “An Open Source Decision Support System Implementation,” p. 2.

[74] A. Farrag, R. Medlin, C. Moore, and J. Mostafa, “An Open Source Decision Support System,” p. 2.

[75] “Drools & jBPM Info Sheet.” [Online]. Available: http://www.slideshare.net/MarkProctor/drools-jbpm-info-sheet.

[76] R. H. J. Sprague, “A framework for the development of decision support systems,” *MIS quarterly*, vol. 4, no. 4, pp. 1–26, 1980.

[77] O. Zimmermann, N. Schuster, and P. Eeles, “Modeling and sharing architectural decisions, Part 1: Concepts,” 05-Aug-2008. [Online]. Available: http://www.ibm.com/developerworks/architecture/library/ar-knowwiki1/.

[78] O. Zimmermann, “Guidance Models and Decision-Making Tooling for SOA, Cloud, and Outsourcing Solution Design,” 2011.

[79] Microsoft, “Transaktionsverarbeitung,” 2010. [Online]. Available: http://msdn.microsoft.com/de-de/library/ee818756.aspx.

[80] N. Leavitt, “Will NoSQL Databases Live Up to Their Promise?,” *Computer*, vol. 43, no. 2, pp. 12–14, Feb. 2010.

[81] S. Edlich, “NOSQL Databases.” [Online]. Available: http://nosql-database.org/.

[82] C. Strauch, “NoSQL Databases.”

[83] R. Ramakrishnan, “CAP and Cloud Data Management,” *Computer-IEEE Computer*, no. February, pp. 43–49, 2012.

[84] “Transactions - Google App Engine — Google Developers.” [Online]. Available: https://developers.google.com/appengine/docs/java/datastore/transactions.

[85] D. Pritchett, “Base: An acid alternative,” *Queue*, no. 3, pp. 48–55, 2008.

[86] D. Song, E. Shi, I. Fischer, and U. Shankar, “Cloud Data Protection for the Masses,” *Development*, no. January, pp. 39–46, 2012.

[87] “Atomic Operations - MongoDB.” [Online]. Available: http://www.mongodb.org/display/DOCS/Atomic+Operations.

[88] “About Writes in Cassandra | DataStax Cassandra 0.8 Documentation.” [Online]. Available: http://www.datastax.com/docs/0.8/dml/about_writes.

[89] “Amazon DynamoDB FAQs.” [Online]. Available: http://aws.amazon.com/de/dynamodb/faqs/#What_does_read_consistency_mean_Why_should_I_care.

[90] “Scaling Web Databases: Auto-Sharding with MySQL Cluster (Oracle’s MySQL Blog).” [Online]. Available: https://blogs.oracle.com/MySQL/entry/scaling_web_databases_auto_sharding.

[91] G. Tan, V. Sreedhar, and G. R. Gao, “Just-In-Time Locality and Percolation for Optimizing Irregular Applications on a Manycore Architecture.”

[92] J. Maassen and T. Kielmann, “Parallel application experience with replicated method invocation,” *Practice and Experience*, pp. 1–31, 2001.

[93] A. Gray, “Amazon Web Services Blog: AWS HowTo: Using Amazon Elastic MapReduce with DynamoDB (Guest Post),” 2012. [Online]. Available: http://aws.typepad.com/aws/2012/01/aws-howto-using-amazon-elastic-mapreduce-with-dynamodb.html.

[94] “Was ist AWS?” [Online]. Available: http://aws.amazon.com/de/what-is-aws/.

[95] “Produkte.” [Online]. Available: http://aws.amazon.com/de/products/.

[96] “Datenbanken in Amazon EC2 ausführen.” [Online]. Available: http://aws.amazon.com/de/running_databases/#relational_amis.

[97] Bruce, “Query 201: Tips and Tricks for Amazon SimpleDB Query,” 2009. [Online]. Available: http://aws.amazon.com/articles/1232.

[98] Amazon Web Services, “Limits - Amazon SimpleDB.” [Online]. Available: http://docs.amazonwebservices.com/AmazonSimpleDB/latest/DeveloperGuide/SDBLimits.html.

[99] “Amazon Relational Database Service (Amazon RDS).” [Online]. Available: http://aws.amazon.com/rds/.

[100] “Microsoft Unveils Windows Azure at Professional Developers Conference: Company releases comprehensive Azure Services Platform for the cloud, offering unprecedented power of choice and open connections for developers.” [Online]. Available: http://www.microsoft.com/presspass/press/2008/oct08/10-27pdcday1pr.mspx.

[101] “Intro to Windows Azure.” [Online]. Available: http://www.windowsazure.com/en-us/develop/net/fundamentals/intro-to-windows-azure/.

[102] “Windws Azure Compute: PaaS | IaaS | Cloud Services.” [Online]. Available: http://www.windowsazure.com/en-us/home/features/compute/.

[103] “Using the Windows Azure Storage Services.” [Online]. Available: http://msdn.microsoft.com/en-us/library/windowsazure/ee924681.aspx.

[104] Google, “Änderungsverlauf Google App Engine,” 2012. [Online]. Available: https://developers.google.com/appengine/docs/revision_history?hl=de.

[105] “Microsoft Case Study: Windows Azure - Flavorus.” [Online]. Available: http://www.microsoft.com/casestudies/Windows-Azure/Flavorus/Ticketing-Company-Scales-to-Sell-150-000-Tickets-in-10-Seconds-by-Moving-to-Cloud-Computing-Solution/4000011072.

[106] M. Fowler, *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2003.

[107] “VM Import.” [Online]. Available: http://aws.amazon.com/ec2/vmimport/.

[108] “Microsoft Case Study: Windows Azure - Festival Internacional Cervantino.” [Online]. Available: http://www.microsoft.com/casestudies/Windows-Azure/Festival-Internacional-Cervantino/Major-Mexican-Cultural-Event-Builds-Cloud-Based-Website-to-Deliver-On-Demand-Media/4000010922.

[109] “Microsoft Case Study: Windows Azure - Socialblaze.” [Online]. Available: http://www.microsoft.com/casestudies/Windows-Azure/Socialblaze/Tech-Startup-Moves-from-Amazon-to-Windows-Azure-Foresees-10-Percent-Revenue-Gain/4000009719.

[110] “Microsoft Case Study: Windows Azure - Gradatim.” [Online]. Available: http://www.microsoft.com/casestudies/Windows-Azure/Gradatim/ISV-Moves-Microfinance-Platform-to-the-Cloud-Shortens-Deployments-by-84-Percent/4000011637.

[111] “Microsoft Case Study: Windows Azure - HRLocker.” [Online]. Available: http://www.microsoft.com/casestudies/Windows-Azure/HRLocker/Software-Vendor-Offers-Cloud-Based-HR-Management-Solution-Lowers-Costs-by-77-Percent/4000009552.

[112] “Microsoft Case Study: Windows Azure - Pixel Pandemic.” [Online]. Available: http://www.microsoft.com/casestudies/Windows-Azure/Pixel-Pandemic/Online-Game-Studio-Minimizes-Costs-Enhances-Margin-with-Cloud-Technology/4000011427.

[113] “Issuu Case Study: Amazon Web Services.” [Online]. Available: http://aws.amazon.com/solutions/case-studies/issuu/.

[114] “PIXNET Case Study: Amazon Web Services.” [Online]. Available: http://aws.amazon.com/solutions/case-studies/pixnet/.

[115] “National Taiwan University Case Study: Amazon Web Services.” [Online]. Available: http://aws.amazon.com/solutions/case-studies/national-taiwan-university/.

[116] “Dropbox - Mach dir das Leben einfacher.” [Online]. Available: https://www.dropbox.com/.

[117] “Eine Welt voller Musik - Spotify.” [Online]. Available: http://www.spotify.com/de/.

[118] “Foodspotting - Find and recommend dishes, not just restaurants.” [Online]. Available: http://www.foodspotting.com/.

[119] “PicTranslator Case Study: Amazon Web Services.” [Online]. Available: http://aws.amazon.com/solutions/case-studies/pictranslator/.

[120] H. P. Luhn, “A Business Intelligence System,” *IBM Journal of Research and Development*, vol. 2, no. 4, pp. 314 – 319, 1958.

[121] “Global Blue Case Study: Amazon Web Services.” [Online]. Available: http://aws.amazon.com/solutions/case-studies/global-blue/.

[122] “Chalklabs Case Study: Amazon Web Services.” [Online]. Available: http://aws.amazon.com/solutions/case-studies/chalklabs/.

[123] “Atbrox and Lingit Case Study: Amazon Web Services.” [Online]. Available: http://aws.amazon.com/solutions/case-studies/atbrox/.

[124] “Running MySQL on Amazon EC2 with EBS (Elastic Block Store) : Articles & Tutorials : Amazon Web Services.” [Online]. Available: http://aws.amazon.com/articles/1663.

[125] “jclouds.” [Online]. Available: http://www.jclouds.org/.

[126] J. Baker, C. Bond, J. Corbett, and J. Furman, “Megastore: Providing scalable, highly available storage for interactive services,” *5th Biennial Conference on Innovative Data Systems Research (CIDR ’11)*, pp. 223–234, 2011.

[127] S. S. Anand, “Netflix ’ s Transition to High-Availability Storage Systems,” *Soccer*, no. October, pp. 1–9, 2010.

[128] “Alexa Case Study: Amazon Web Services.” [Online]. Available: http://aws.amazon.com/en/solutions/case-studies/alexa/.

[129] Canuckguy et al., “Weltkarte.” [Online]. Available: http://de.wikipedia.org/wiki/Datei:BlankMap-World6,_compact.svg.

[130] “Amazon CloudFront.” [Online]. Available: http://aws.amazon.com/cloudfront/.

[131] “How to Shard with SQL Azure - TechNet Articles - United States (English) - TechNet Wiki.” [Online]. Available: http://social.technet.microsoft.com/wiki/contents/articles/1926.how-to-shard-with-sql-azure.aspx.

[132] “Using an Example of Sharding with Hibernate : Articles & Tutorials : Amazon Web Services.” [Online]. Available: http://aws.amazon.com/articles/0040302286264415.

[133] *GRAMM–LEACH–BLILEY ACT*. 1999, pp. 1–145.

[134] *Sarbanes-Oxley Act of 2002*. 2002.

[135] “Microsoft Case Study: Windows Azure - Datacastle.” [Online]. Available: http://www.microsoft.com/casestudies/Windows-Azure/Datacastle/Software-Company-Helps-Customers-Secure-Data-with-Scalable-Cloud-Solution/4000008339.

[136] Consultative Committee for Space Data Systems, “Reference Model for an Open Archival Information System (OAIS),” 2002.

[137] “Archive on AWS.” [Online]. Available: http://aws.amazon.com/archive/.

[138] “Open Data Network | Netzwerk zur Förderung von Open Government, Open Data, Transparenz und Partizipation.” [Online]. Available: http://opendata-network.org/.

[139] “Streaming API Methods | Twitter Developers.” [Online]. Available: https://dev.twitter.com/docs/streaming-api/methods.

[140] M. Mathioudakis and N. Koudas, “TwitterMonitor: Trend Detection over the Twitter Stream,” *SIGMOD ’10 Proceedings of the 2010 international conference on Management of data*, pp. 1155–1157, 2010.

[141] S. Das, S. Nishimura, D. Agrawal, and A. E. Abbadi, “Albatross : Lightweight Elasticity in Shared Storage Databases for the Cloud using Live Data Migration,” vol. 4, no. 8, pp. 494–505, 2011.

[142] “Accelerating Application Development using Amazon RDS.” [Online]. Available: http://event.on24.com/eventRegistration/EventLobbyServlet?target=lobby.jsp&eventid=450110.

[143] M. Keep and A. Morgan, “Scaling Web Databases: MySQL Cluster. Auto-Partitioning, SQL & NoSQL Interfaces, Schema Flexibility.”

[144] “Global Infrastructure.” [Online]. Available: http://aws.amazon.com/de/about-aws/globalinfrastructure/.

[145] “Using Regions and Availability Zones - Amazon Elastic Compute Cloud.” [Online]. Available: http://docs.amazonwebservices.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html.

[146] Amazon Web Services, “AWS Import/Export.” .

[147] Microsoft, “Multi-Tenant Data Architecture.” [Online]. Available: http://msdn.microsoft.com/en-us/library/aa479086.aspx.

[148] “Using Hibernate to Implement Multi-tenant Cloud Architecture.” [Online]. Available: http://www.devx.com/Java/Article/47817/1954.

[149] Google, “Implementieren der Mehrinstanzenfähigkeit mithilfe von Namespaces - Google App Engine — Google Developers.” [Online]. Available: https://developers.google.com/appengine/docs/java/multitenancy/multitenancy?hl=de.

[150] Amazon Web Services, “Amazon Virtual Private Cloud.” [Online]. Available: http://aws.amazon.com/de/vpc/.

[151] Services Web Services, “Using DynamoDB with Amazon Elastic MapReduce : Articles & Tutorials : Amazon Web Services.” [Online]. Available: http://aws.amazon.com/articles/28549.

[152] Google, “Importing and Exporting Data - Google Cloud SQL.” [Online]. Available: https://developers.google.com/cloud-sql/docs/import_export.

[153] Amazon Web Services, “WM Import/Export - Amazon Web Services.” [Online]. Available: http://aws.amazon.com/ec2/vmimport/.

[154] “Deltacloud - Documentation.” [Online]. Available: http://deltacloud.apache.org/drivers.html#h2.

[155] SNIA, “SNIA Europe - About CDMI,” 2011. [Online]. Available: http://www.snia-europe.org/en/technology-topics/cloud-storage/about-cdmi.cfm.

Alle Links wurden am 5. September 2012 letztmalig überprüft.

Erklärung

Hiermit versichere ich, diese Arbeit selbstständig verfasst und nur die angegebenen Quellen benutzt zu haben.

Stuttgart, 05.09.12 _____

(Thomas Bachmann)