Kantonsspital Graubünden
Departement ICT
Data Center
www.ksgr.ch

Michael
Graber
Datenbank Administrator
michael.graber@ksgr.ch
+41 81 256 68 25

Diplomarbeit Technik und Wirtschaftsinformatik 2023-2024

# PostgreSQL HA Cluster - Konzeption und Implementation

22.11.2023

## Management Summary

Diplomarbeit Michael Graber

# Inhaltsverzeichnis

## Abkürzungen

| | |
|---|---|
| ICT | information and communications technology |
| ibW | ibW Höhere Fachschule Südostschweiz |
| KSGR | Kantonsspital Graubünden |
| RDBMS | Relational Database Management System |
| DBMS | Database Mananagement System |
| k8s | Kubernetes |
| HPE | Hewlett Packard Enterprise |
| HP-UX | Hewlett Packard UNIX |
| SAP | Systemanalyse Programmentwicklung |
| SQL | Structured Query Language |
| DBA | Database Administrator / Datenbankadministrator |
| HA | High Availability |
| PRTG | Paessler Router Traffic Grapher |
| SAN | Storage Area Network |
| SIEM | Security Information and Event Management |
| CI/CD | Continuous Integration/Continuous Delivery |
| SWOT-Analyse | Strengths, Weaknesses, Opportunities, Threats |
| OLAP | Online Analytical Processing |
| IaC | Infrastructure as Code |
| IPERKA | Informieren, Planen, Entscheiden, Realisieren, Kontrollieren, Auswerten |
| BSI | Bundesamt für Sicherheit in der Informationstechnik |
| VRRP | Virtual Router Redundancy Protocol |

# 1 Einleitung

## 1.1 Ausgangslage und Problemstellung

### 1.1.1 Das Kantonsspital Graubünden

Das Kantonsspital Graubünden ist das Zentrumsspital der Südostschweiz, welches Teil der sogenannten Penta Plus Spitäler ist. Die Penta plus Spitäler sind das Kantonsspital Baden, das Kantonsspital Winterthur, das Spitalzentrum Biel AG, das Kantonsspital Baselland, die Spital STS (Simmental-Thun-Saanenland) AG und eben das Kantonsspital Graubünden.
Das KSGR deckt dabei die Spitalregion Churer Rheintal ab

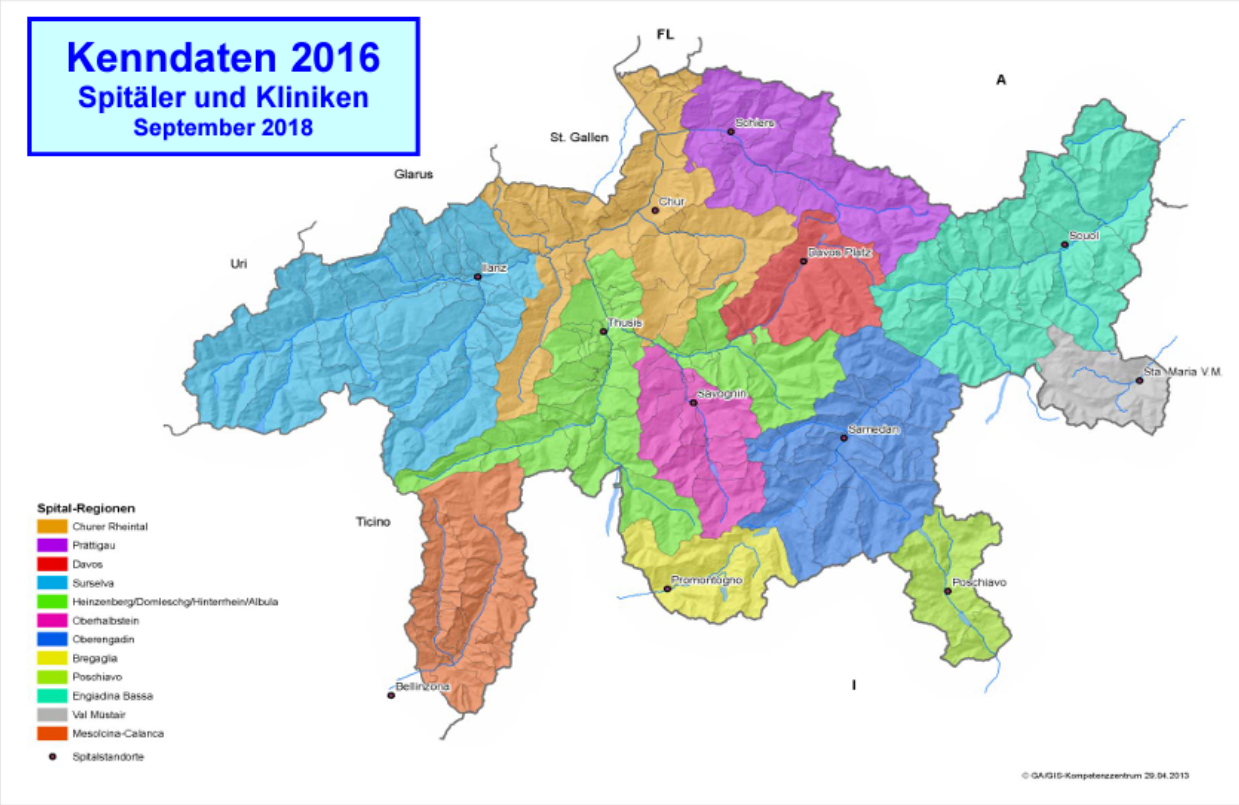


Abbildung 1.1: Spitalregionen Kanton Graubünden[20]

Seit dem 1. Januar 2023 betreibt das KSGR den Standort Walenstadt im Kanton St. Gallen und deckt primär den Wahlkreis Sarganserland ab.


Abbildung 1.2: Wahlkreise Kanton St. Gallen[41]

Da dieser Wahlkreis der Spitalregion Rheintal Werdenberg Sarganserland zugeordnet ist, wird das KSGR auch im restlichen südlichen Teil der Spitalregion aktiv sein.


Abbildung 1.3: Spitalregionen / Spitalstrategie Kanton St. Gallen[14]

### 1.1.2 Die ICT des Kantonsspital Graubünden

Das Kantonsspital Graubünden hat eine Matrixorganisation. Die ICT ist ein eigenständiges Departement und gilt als sogenanntes Querschnittsdepartement, dh. die ICT bedient alle anderen Departemente.

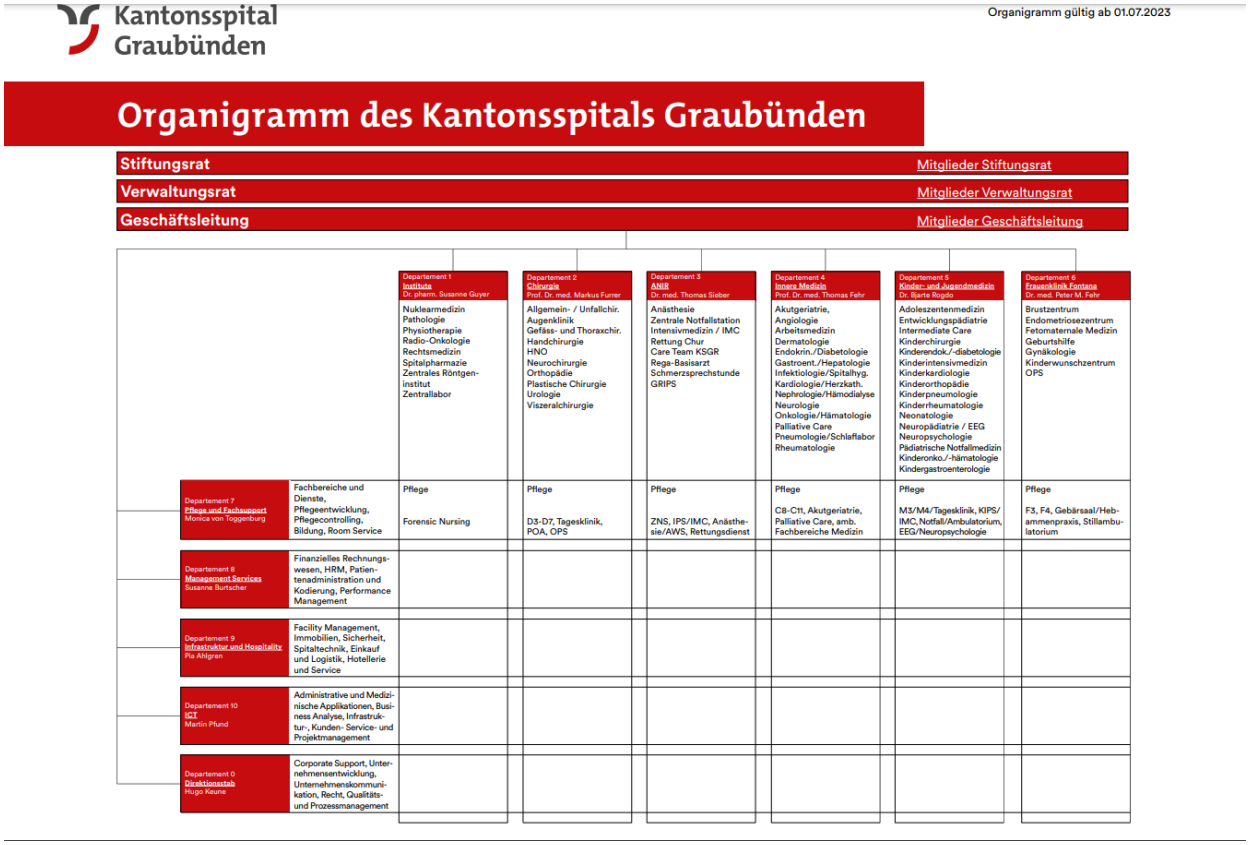


Abbildung 1.4: Organigramm Kantonsspital Graubünden

Die ICT betreibt über 400 Applikationen die auf mehr als 1055 physische und virtuelle Server und Appliances. Das Rückgrat der Infrastruktur ist dabei die Virtualisierungsplattformen VMware ESXi für Server und Citrix für die Thinclients der Enduser. Es werden aber auch Dienstleistungen für andere Spitäler und Kliniken oder andere Einrichtungen des Gesundheitswesens erbracht. Entsprechen wurde die ICT in ein Applikationsmanagement, ein Infrastrukturmanagement sowie einem unterstützenden Bereich aufgegliedert. Das Applikationsmanagement wurde in je einen Bereich für die Administrativen und Medizinischen Applikationen aufgeteilt. Das Infrastrukturmanagement wiederum wurde in den Bereich Netzwerk und Data Center, welcher für Server zuständig ist, aufgeteilt. Der Bereich Business- und Prozessunterstützung beinhaltet je eine Abteilung für die Businessanalyse, das Projektmanagement und Benutzer- und Clientservices in der auch der Service-Desk untergebracht ist.

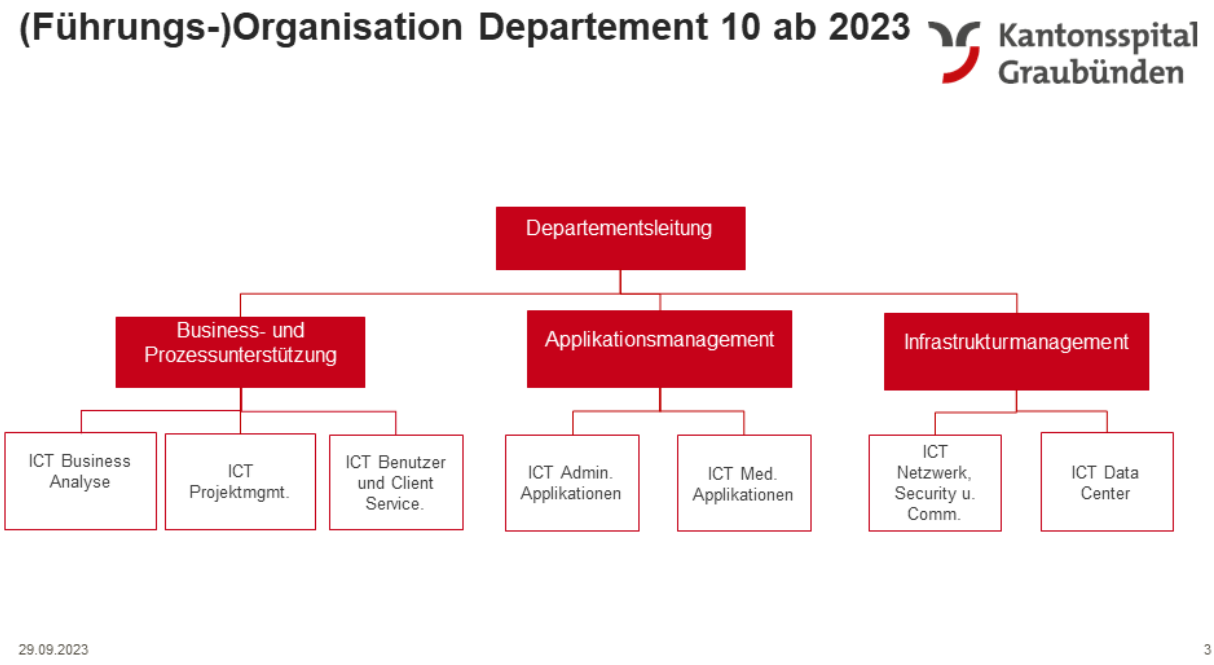


Abbildung 1.5: Organigramm Departement 10 - ICT

Die Organisation der ICT wird sich aber bis spätestens zum Abschluss der Diplomarbeit noch verändern.

### 1.1.3 Rolle in der ICT vom Kantonsspital Graubünden

Meine Rolle im Kantonsspital Graubünden resp. in der ICT ist die eines DBA. Diese Rolle ist in der Abteilung ICT Data Center.
Da die Kernsysteme auf Oracle Datenbanken und HP-UX laufen, bin ich primär Oracle Database DBA und manage das HP-UX in Zusammenarbeit mit HPE. Die administrative Tätigkeit bei HP-UX besteht primär im Betrieb der HP-UX Cluster Packages (einer sehr rudimentären Art von Containern), überwachen und erweitern des Filesystems, erweitern von SAN Storage Lunes für die Filesystem erweiterung, Erstellen von PRTG-Sensoren für das Monitoring, SAP Printerqueue Management und andere Tasks die es noch auszuführen gibt. Daneben bin ich auch für andere Datenbanken, teilweise aber nur begrenzt Microsoft SQL Server, MySQL / MariaDB und vermehrt PostgreSQL zuständig. Darüber hinaus bin ich Teilweise in die Linux-Administration involviert und betreue auch noch einige Windows Server für das Zentrale klinische Informationssystem.

### 1.1.4 Ausgangslange

Die meisten der über 400 Applikationen, die das KSGR betreibt, haben in den allermeisten Fällen ihre Daten in Datenbanksysteme speichern. Entsprechend der Vielfalt der Applikationen existieren auch eine vielzahl an Datenbanksystemen und Versionen.
Basierend auf der Liste *DB-Engines Ranking*[11] der Top-Datenbanksysteme . Allerdings werden nicht alle Datenbanksysteme berücksichtigt, entweder weil das Datenbanksystem keine Client/Server Architektur hat oder nicht im Scope der IT oder des Projekts ist.
Folgende Datenbanken sind inventarisiert:

| DBMS | Datenbankmodell | Inventarisiert | Kommentar |
|---|---|---|---|
| Oracle Database | Relational, NoSQL, OLAP | Ja | |
| MySQL | Relational | Ja | |
| Microsoft SQL Server | Relational, NoSQL, OLAP | Nein | Werden separat administriert und sind daher nicht in diesem Inventar gelistet |
| PostgreSQL | Relational, NoSQL | Ja | |
| MongoDB | NoSQL | Ja | |
| Redis | Key-value | Ja | |
| Elasticsearch | Search engine | Ja | |
| IBM DB2 | Relational | Ja | |
| SQLite | Relational | Nein | Lokale Datenbank. Zudem wird die DB nicht via Netzwerk angesprochen |
| Microsoft Access | Relational | Nein | Nicht im Scope der ICT |
| Snowflake | Relational | Ja | |
| Cassandra | Relational | Ja | |
| MariaDB | Relational | Ja | |
| Splunk | Search engine | Ja | |
| Microsoft Azure SQL Database | Relational, NoSQL, OLAP | Nein | Datenbanken sind nicht On-Premise und somit nicht im Scope |

Tabelle 1.1: Inventarisierte Datenbanksysteme

Folgende Datenbanksysteme sind demnach im KSGR im Einsatz:

| RDBMS | Summe RDBMS / Cluster / CDB / Instance | Summe Databases |
|---|---|---|
| MariaDB | 2 | 2 |
| MongoDB | 2 | 2 |
| MySQL | 28 | 50 |
| Oracle Database | 27 | 30 |
| PostgreSQL | 20 | 20 |
| Redis | 1 | 1 |
| **Gesamtergebnis** | **80** | **105** |

Tabelle 1.2: Datenbankinventar

Aufgeschlüsselt auf die Betriebssysteme auf denen die Datenbanken laufen, ergibt sich folgendes Bild:

| OS / RDBMS | Summe RDBMS / Cluster / CDB / Instance | Summe Databases |
|---|---|---|
| **HP-UX** | **21** | **24** |
| Oracle Databases | 21 | 24 |
| **Linux** | **26** | **48** |
| MariaDB | 2 | 2 |
| MySQL | 14 | 36 |
| Oracle Database | 1 | 1 |
| PostgreSQL | 8 | 8 |
| Redis | 1 | 1 |
| **Windows Server** | **33** | **33** |
| MongoDB | 2 | 2 |
| MySQL | 14 | 14 |
| Oracle Databases | 5 | 5 |
| PostgreSQL | 12 | 12 |
| **Gesamtergebnis** | **80** | **105** |

Tabelle 1.3: Datenbankinventor - Nach Betriebssystemen aufgeschlüsselt

Die Kernsysteme des Spitals werden auf Oracle Datenbanken (Oracle Database)betrieben, die aktuell auf einer HP-UX betrieben werden. Stand heute gibt es kein Clustersystem für die Open-Source Datenbanken wie MariaDB/MySQL oder PostgreSQL.
Durch die Einführung von Kubernetes als Containerplattform wird der Bedarf an PostgreSQL

Datenbanken immer grösser. Es werden in naher Zukunft auch verschiedene Oracle Datenbanken sowie MySQL Datenbanken auf PostgreSQL migriert werden.
Aktuell werden die Daten des Zabbix der Netzwerktechniker auf eine MariaDB Datenbank gespeichert, dies soll sich aber ändern. Da das Zabbix alle Netzwerkgeräte Überwacht, pro Sekunde werden im Moment 1'200 Datenpunkte abgefragt und xxx in die Datenbank und wird im Laufe der Zeit mehrere Terrabyte gross werden.

### 1.1.5 Problemstellung

Zusammen mit den bestehenden PostgreSQL-Datenbankinstanzen werden die PostgreSQL Datenbanken in der Art, wie sie bisher Betrieben werden, nicht mehr Betreibbar sein. Die bisherige Strategie erzeugt sehr viele Aufwände und provoziert Risiken, namentlich:

- dezentrale Backups und fragmentierte Backup-Strategien
  - Fehlende Kontrolle
  - Wiederherstellbarkeit nicht garantiert
- Verschiedene Betriebssysteme mit verschiedenen Versionen
  - Fehlernder Überblick
  - Veraltete Betriebssystem- und Datenbankversionen
  - Grosser Administrationsaufwand
- Uneinheitliche Absicherung und Härtung
  - Hohe Angreifbarkeit
  - Veraltete Betriebssystem- und Datenbankversionen
  - Grosser Administrationsaufwand
- Uneinheitliche HA-Fähigkeit
  - Hohe Angreifbarkeit
  - Veraltete Betriebssystem- und Datenbankversionen
  - Grosser Administrationsaufwand

Dadurch ergeben sich nach BSI folgende Risiken:

| Identifikation | | | | | | Abschätzung | | Behandlung | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ID | Schutzziel | Referenz BSI 200-3 | Risiko | Beschreibung / Ursache | Auswirkung | WS | SM | Massnahmen ergreifen? | Zielwert WS | Zielwert SM | Massnahme |
| 1 | I | G0.22 | Manipulation von Informationen | Durch veraltete Systeme die zudem unterschiedlich gut gehärtet und gesichert sind (z.B. durch Verschlüsselung des Verkehrs oder der Daten auf dem Storage), besteht das Risiko das Daten manipuliert werden | Die Auswirkungen reichen von einer Fehlfunktion des Systems bis hin zum vollständigen Verlust der Integrität der Daten | 2 | 4 | Ja | 1 | 2 | Best-Practice bei Härtung der Systeme.<br>Redundanzen einführen |
| 2 | A | G0.25 | Ausfall von Geräten oder Systemen | Manche Datenbanken und deren Betriebssysteme sind sehr alt und sehr lange im Einsatz.<br>Einige dieser Systeme sind schon so alt, das keine Hotfixes, Patches und Updates mehr erhältlich sind.<br>Hierdurch entsteht das Risiko, das Systeme Ausfallen | Sofern keine HA-Architektur aufgebaut wurde, ist die Verfügbarkeit ernsthaft gefährdet resp. die Applikation steht nicht mehr zur Verfügung. | 4 | 4 | Ja | 2 | 2 | Redundanzen einführen |
| 3 | C, I, A | G0.26 | Fehlfunktion von Geräten oder Systemen | Manche Datenbanken und deren Betriebssysteme sind sehr alt und sehr lange im Einsatz.<br>Einige dieser Systeme sind schon so alt, das keine Hotfixes, Patches und Updates mehr erhältlich sind.<br>Hierdurch entsteht das Risiko, das Systeme Fehlfunktionen erleiden.<br><br>Allerdings versuchen Datenbanksysteme, die Auswirkungen so gering wie möglich zu halten. | Fehlfunktionen können innerhalb von Datenbanksystemen die Datenkonsistenz verletzen, Daten können verloren gehen oder ungewollt von Dritten und unberechtigten Personen eingesehen werden. Systeme könnten nicht mehr oder nur noch eingeschränkt verfügbar werden.<br><br>Daher sind sowohl Vertraulichkeit, Integrität und Verfügbarkeit gefährdet. | 2 | 4 | Ja | 2 | 2 | Systeme zentralisieren<br>Lifecycle etablieren |
| 4 | C, I, A | G0.27-1 | Ressourcenmangel (personelle Ressourcen) | Aufgrund der sehr heterogenen Landschaft ist der Administrationsaufwand für die jetzigen Systeme sehr gross.<br>Zu gross, als das für jede Datenbank und deren Betriebssystem die notwendige Zeit für eine bedarfsgerechte Administration erbracht werden kann.<br><br>Dadurch bleiben Fehler länger unentdeckt, Hotfixes, Patches, Updates und Upgrades können nicht oder nicht zur richtigen Zeit eingespielt werden.<br><br>Bei einem akuten Problemfall ist nicht garantiert, das die Leute erreichbar sind, die notwendig sind | Die Auswirkungen können vielfältig sein, abhängig davon welcher Aspekt unter dem Ressourcenmangel leidet.<br><br>Grundsätzlich wird aber sowohl die Vertraulichkeit, Integrität und Verfügbarkeit gefährdet. | 3 | 3 | Ja | 2 | 3 | Systeme zentralisieren |
| 5 | A | G0.27-2 | Ressourcenmangel (technische Ressourcen) | Kann auftreten wenn Ressorucenwachstum zu spät bemerkt wird.<br>So kann die CPU Usage oder das Memory Usage schnell anwachsen.<br>Auch der Storage eines Betriebssystems kann nicht mehr ausreichend für ein System werden. | Wenn die CPU- und Memory-Usage über einen gewissen Schwellwert geht, fängt das Betriebssystem an zu Priorisieren.<br>Dies wird primär der Endanwender in form von Performance Einbussen bemerken.<br>Im schlimmsten Fall steht eine Anwendung nicht mehr zur Verfügung.<br><br>Gefährlicher sind Storage Overflows, besonders wenn die Datenbank nicht mehr alle Informationen schreiben konnte, die sie für einen korrekten Neustart benötigte.<br><br>Doch die folgen bleiben nichtsdesto trotz überschaubar.<br>Abhängig davon, welche Fehler gemacht wurden können die Auswirkungen auch stark variieren. | 2 | 2 | Ja | 1 | 2 | Monitoring verschärfen |
| 6 | C, I, A | G0.31 | Fehlerhafte Nutzung oder Administration von Geräten und Systemen | Durch die Vielfalt an Datenbankversionen und Betriebssystemen und Plattformen worauf diese betrieben werden, besteht allen voran das Risiko einer Fehlerhafter Administration und Konfiguration. | Sie reichen von fehlender Verschlüsselung bis hin zu nicht vorhandenem Backup mit nicht mehr gesicherter Wiederherstellbarkeit von Systemen.<br><br>Daraus erschliesst sich das auch bei diesem Risiko die Vertraulichkeit, Integrität und Verfügbarkeit gefährdet ist. | 4 | 3 | Ja | 2 | 3 | Systeme zentralisieren |
| 7 | C, I, A | G0.32 | Missbrauch von Berechtigungen | Obwohl das Microsoft Active Directory die Zentrale Benutzerverwaltung ist, sind die wenigsten Datenbanken an dieses angeschlossen.<br>Hinzu kommt der umstand, das in der Vergangenheit jeder Softwarelieferant sein eigenes Benutzerkonzept mitgebracht hat, auch bei den Datenbankzugängen.<br><br>Multipliziert mit der Anzahl der unterschiedlichsten Datenbanken, Betriebssystemen und Applikationen entsteht das Risiko, das Berechtigungen Wissendlich oder Unwissendlich missbraucht werden.<br>Verschiedene Datenbanken sind Standalone Cluster (Instanzen) welche über keinen Failover-Mechanismus verfügen. | Der Wissentliche oder Unwissentliche Missbrauch von Berechtigungen kann verheerende Auswirkungen haben.<br>Unter anderem können Daten missbräuchlich abgezogen werden, Daten manipuliert oder das ganze System komplett zerstört werden. | 2 | 4 | Ja | 2 | 2 | Systeme zentralisieren<br><br>Übergreifendes Berechtigungskonzept einführen<br>Monitoring der Zugriffe |
| 8 | A, I | G0.45 | Datenverlust | Zudem wurden die meisten Datenbanken nur mittels Snapshots oder einem Filesystem Backup gesichert, nicht über eine eigentliche Sicherung mittels WAL.<br>Gerade die fehlende WAL-Archivierung führt im Backupfall dazu, das alle Transaktionen die zwischen dem letzten Backup nicht mehr vorhanden sind.<br><br>Hinzu kommt, das für die meisten Datenbanken hohe Sicherungsintervalle von einmal pro Stunde oder gar nur einmal am Tag gewählt wurde.<br><br>Ein weiterer Aspekt des Risikos besteht in der tatsache, das aufgrund der grossen Anzahl Datenbanken und deren Heterogenität nur wenige Backups auch wirklich regelmässig geprüft werden. | Aus dem Risiko ergeben sich zwei Auswirkungen, die aber beide ein hohes Mass an Schaden verursachen können.<br><br>Erstens könnten Backups gar nicht mehr Wiederhergestellt werden, dies hätte dann einen Totalen Datenverlust zur Folge.<br>Die zweite Ursache erwächst auf der fehlenden WAL-Archivierung, dadurch können zwar die Daten bis zu einem Zeitpunkt X Wiederhergestellt werden allerdings sind diese dann nicht zwingend Konsistent. | 4 | 5 | Ja | 1 | 3 | Systeme zentralisieren<br>Einheitliches Backupkonzept<br>Regelmässige Restore-Tests |

Tabelle 1.4: Risiko-Matrix aktuelle Situation PostgreSQL Datenbanken

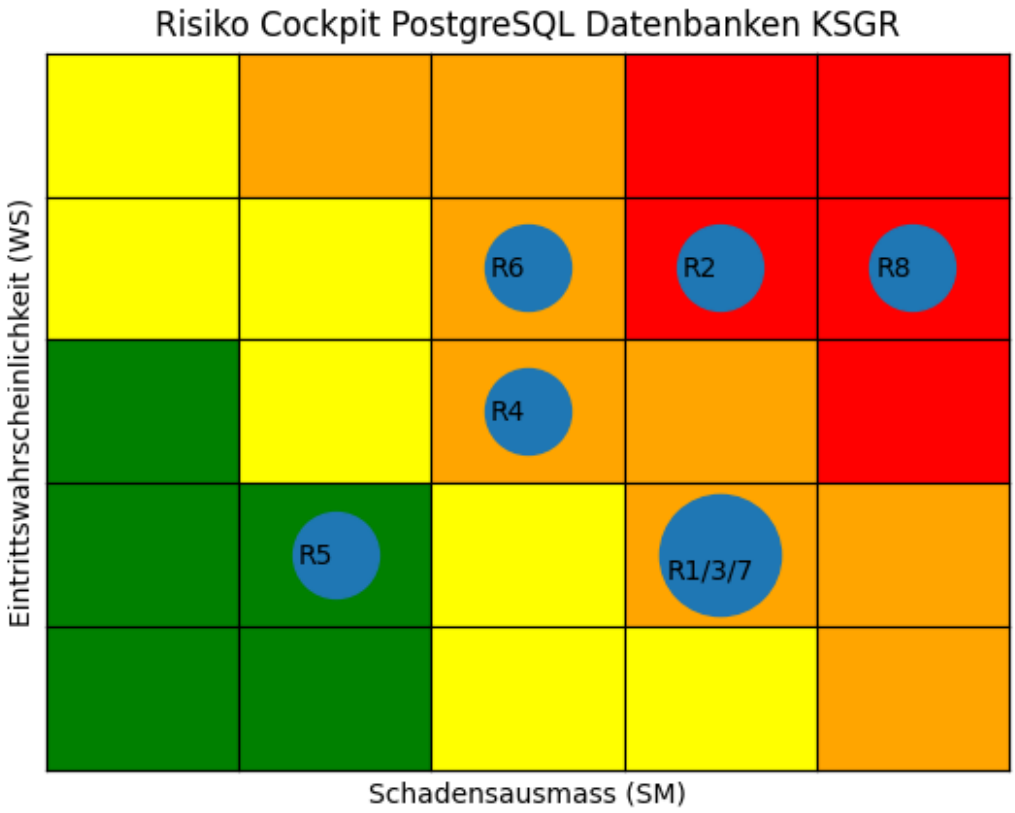


Abbildung 1.6: Risiken bestehende Lösung

Daraus ergeben sich folgende Strategien und Handlungsfelder um die Massnahmen zur Risikominimierung umzusetzen:

- Systemabsicherung erarbeiten und einsetzen
- HA-Clustering einführen um die Redundanz zu gewährleisten und Systeme zentral verwalten und betreiben zu können
- Lifecycle-management für Datenbanken und Betriebssysteme erarbeiten und einsetzen
- Backupkonzept erarbeiten
- Berechtigungskonzept erarbeiten und einführen

Mit diesen Massnahmen lassen sich die Risiken gesenkt werden:

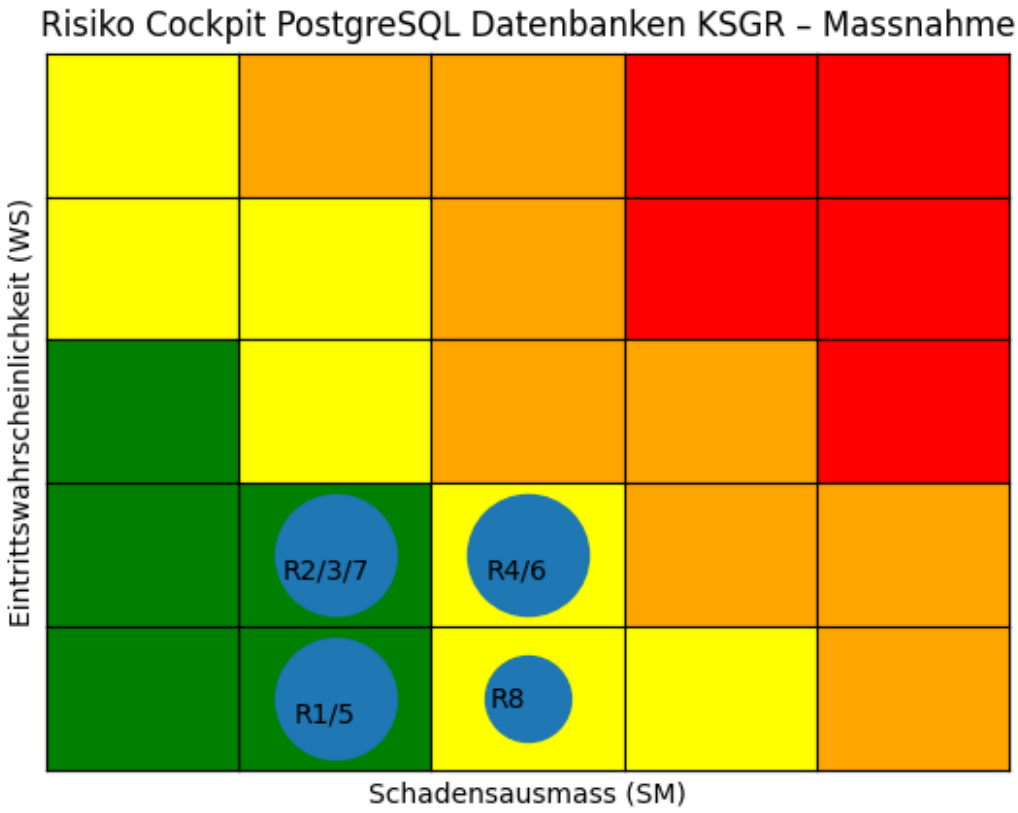


Abbildung 1.7: Risiken bestehende Lösung mit Massnahmen

## 1.2 Zieldefinition

Das administrieren einer PostgreSQL Datenbank umfasst i.d.R. [28, 33] folgende zehn Tasks die zum täglichen Alltag gehören:

| Nr. | Aufgabe | Beschreibung | Wichtigkeit |
|---|---|---|---|
| 1 | Failover | In einem Fehlerfall soll die DB-Node auf einen Standby-Node übergeben werden.<br>Nach einem Failover muss der DB-Node wieder vom Standby-Node auf den Primären Node zurückgesetzt werden. | Hoch |
| 2 | Failover Restore | Dabei darf es zu keinem Datenverlust kommen, also alle Daten die auf dem Standby-Node erfasst wurden,<br>müssen auf den Primären DB-Node zurückgeschrieben werden beim Failover Restore | Hoch |
| 3 | Filesystem Management | Die Datenmenge von Datenbanken wachsen in der Regel beständig.<br>Die Belegung von Tablespaces und Filesystem muss deshalb Überwacht und ggf. erweitert werden.<br>Läuft eine Disk voll kommt es im besten Fall zu einem Stillstand der DB, im schlimmsten Fall zu Inkonsistenzen und Datenverlust | Hoch |
| 4 | Monitoring | Nebst den allgemeinen Metriken wie CPU / Memory Usage und der Port Verfügbarkeit gibt es noch eine reihe weiterer Aspekte die Überwacht werden müssen.<br>Zum Beispiel ob es zu verzögerungen bei der Replikation kommt oder die Tablespaces genügend Platz haben.<br>Dazu gehört auch das Überwachen des Logs und entsprechende Schritte im Fehlerfall.<br>PostgreSQL sammelt Statistiken um SQL Queries optimaler ausführen zu können. | Mittel |
| 5 | Statistiken / Cleanup Jobs justieren | Zudem wird im Rahmen des gleichen Sheduled Tasks ein Cleanup Vorgenommen,<br>so dass z.B. gelöschte Datensätze den Disk Space nicht sinnlos belegen.<br>Die Konfiguration dieser Jobs muss an der Metrik der Datenbank angepasst werden,<br>weil gewisse Tasks dann entweder viel zu oft oder viel zu wenig bis gar nicht mehr ausgeführt werden. | Mittel |
| 6 | SQL optimierungen | In PostgreSQL können inperfomante SQL Statements ausgelesen werden und zum Teil werden auch informationen zum Tuning geliefert[9].<br>Diese müssen Regelmässig ausgelesen werden | Tief |
| 7 | Health Checks und Aktionen (Maintenance) | Regelmässig muss die Gesundheit der DBs überprüft werden, etwa ob Tabellen und/oder Indizes sich aufgebläht haben oder ob Locks vorhanden sind[2].<br>Während der Hauptarbeitszeit muss dies mindestens alle 90 Minuten geprüft und ggf. reagiert werden. | Hoch |
| 8 | Housekeeping | Mit Housekeeping Jobs werden regelmässig Trace- und Alertlogfiles aufgeräumt,<br>um Platz auf den Disken zu sparen aber auch um die Übersichtlichkeit zu wahren. | Mittel |
| 9 | Verwalten von DB Objekten | Regelmässig müssen DB Objekte wie Datenbanken, Tabellen, Trigger, Views etc. angepasst oder erstellt werden.<br>Dies richtet sich nach den Bedürfnis der Kunden resp. deren Applikationen. | Tief |
| 10 | User Management | Die Zugriffe der User müssen Überwacht, angepasst, erfasst oder gesperrt werden.<br>Auch diese Aufgabe richtet sich nach den Bedürfnissen der Kunden. | Tief |

Tabelle 1.5: Administrative Aufgaben

Von diesen Tasks müssen Teile davon zu 50% automatisiert werden wobei alle Muss-Aufgaben automatisiert werden müssen. Diese wären nachfolgende Tasks die automatisiert werden können.

| Nr. | Aufgabe | Wichtigkeit | Zu automatisierender Task | Priorität | Muss / Kann | Spätester Termin |
|---|---|---|---|---|---|---|
| 1 | Failover | Hoch | Automatisierter Failover auf mindestens einen Sekundären DB-Node | 1 | Muss | Abgabe |
| 2 | Failover Restore | Hoch | Sobald der Primäre DB-Node wieder vorhanden ist,<br>muss automatisch auf den Primären DB-Node zurückgesetzt werden. | 1 | Muss | |
| 3 | Filesystem Management | Hoch | Das Filesystem muss beim erreichen von 95% Usage automatisiert vergrössert werden.<br>Die Vergrösserung muss anhand der Wachstumsrate (die mittels Linux Commands zu ermitteln ist),<br>vergrössert werden | 4 | Kann | |
| 4 | Monitoring | Mittel | Der Status der Clusterumgebung und der Replikation muss im PRTG überwacht werden | 2 | Muss | |
| 5 | Statistiken / Cleanup Jobs justieren | Mittel | Regelmässig müssen die Parameter für den AUTOVACUUM Job berechnet werden und<br>das Configfile postgresql.conf automatisch angepasst werden<br>Es gibt SQL Abfragen, mit dem fehlende Indizes ermittelt werden können.<br>Diese Indizes sollen automatisiert erstellt werden. | 2 | Muss | |
| 6 | SQL optimierungen | Tief | Im gleichen Zug sollen aber auch Indizes, welche nicht verwendet werden, entfernt werden.<br>Sie tragen nicht nur nichts zu performanteren Abfragen bei<br>sondern beziehen unnötige Ressourcen bei Datenmanipulationen[9]. | 2 | Muss | |
| 7 | Health Checks und Aktionen<br>(Maintenance) | Hoch | Tabellen und Indizes können sich aufblähen (bloaded table / bloaded index)<br>Ist ein Index aufgebläht, kann dies mittels eines REINDEX<br>mit geringem Impact auf die Datenbank gelöst werden[2]. | 2 | Muss | |
| 8 | Housekeeping | Mittel | Log Rotation muss aktiviert werden und alte Logs regelmässig gelöscht werden. | 3 | Muss | |
| 9 | Verwalten von DB Objekten | Tief | Keine automatisierung möglich | 5 | | |
| 10 | User Management | Tief | Regelmässige Reports sollen User aufzeigen, die seit mehr als einer Woche nicht mehr aktiv waren. | 4 | Kann | |

Tabelle 1.6: Automatisierung Administrativer Aufgaben

Mit der Arbeit sollen folgende Ergebnisse und Resultate erzielt werden:

- Ergebnisse
  Mindestens drei Methoden einen PostgreSQL Cluster aufzubauen müssen analysiert und evaluirt werden
- Resultate
  Aus den mindestens drei Methoden muss die optimale Methode ermittelt werden.
  Am Ende muss zudem ein Funktionierendes Testsystem bestehen.

Daraus ergeben sich folgende Ziele:

| Nr. | Ziel | Beschreibung | Priorität |
|---|---|---|---|
| 1 | Evaluation | Am Ende der Evaluationsphase müssen mindestens drei Methoden für einen PostgreSQL HA Cluster müssen evaluirt werden.<br>Innerhalb der evaluation muss analysiert werden, welche Methode oder welches Tool sich hierfür eignen würde. | Hoch |
| 2 | Testsystem | Am Ende der Diplomarbeit muss ein funktionierendes Testsystem Installiert sein. | Hoch |
| 3 | Automatisierter Failover | Ein PostgreSQL Cluster muss im Fehlerfall auf mindestens einen Standby-Node umschwenken.<br>Dabei muss das Timeout so niedrig sein, dass Applikationen nicht auf ein Timeout laufen. | Hoch |
| 4 | Automatisierter Failover Restore | Nach einem Failover muss es zu einem Fallback oder Failover Restore kommen, sobald der Primary-Node wieder verfügbar ist. | Hoch |
| 5 | Monitoring - Cluster Healthcheck | Die wichtigsten Parameter für das Monitoring des PostgreSQL Clusters (isready, Locks, bloaded Tables),<br>der Replikation (Replay Lag, Standby alive) und des PostgreSQL HA Clusters müssen Überwacht werden. | Mittel |
| 6 | AUTOVACUUM - Parameter verwalten | Täglich müssen die Parameter für den AUTOVACUUM Job berechnet werden und<br>das Configfile postgresql.conf automatisch angepasst werden | Mittel |
| 7 | SQL optimierungen - Indizes tracken und verwalten | Täglich fehlende Indizes automatisiert erstellen und nicht mehr verwendete Indizes automatisiert entfernen | Mittel |
| 8 | Maintenance - Indizes säubern | Täglich bloaded Indices, also aufgeblähte Indizes, automatisiert erkennen und mittels REINDEX bereinigen | Hoch |
| 9 | Housekeeping - Log Rotation | Die Log Rotation muss aktiviert werden. Die Logs müssen aber auch in das KSGR-Log Repository geschrieben werden | Hoch |
| 10 | User Management - Monitoring | Nicht verwendete User sollen einmal pro Woche automatisiert erkannt und in einem Report gemeldet werden. | Tief |
| 11 | Evaluationsziel | Am Ende der Evaluationsphase muss ein Entscheid getroffen worden sein, welche Methode verwendet wird. | Hoch |
| 12 | Installationsziel | Die Testinstallation muss Lauffähig sein und zudem alle Anforderungen und Ziele (3 und 4) erfüllen | Hoch |
| 13 | Testziele | Folgende Testziele müssen erreicht werden:<br>1. Der PostgreSQL Cluster muss immer Lauffähig sein solange noch ein Node up ist, unabhängig davon welche Nodes des PostgreSQL HA Clusters down ist<br>2. Ein Switchover auf alle Secondary Nodes muss möglich sein<br>3. Der Fallback auf den Primary Node muss Erfolgreich sein, unabhängig davon ob ein Failover oder Switchover stattgefunden hat<br>4. Das Timeout bei einem Failover / Switchover muss unterhalb der Default Timeouts der Applikationen GitLab und Harbor liegen.<br>5. Das Replay Lag zwischen Primary und Secondary darf beim Initialen Start nicht über eine Minute dauern oder 1KiB nicht überschreiten | Hoch |

Tabelle 1.7: Ziele

## 1.3 Abgrenzung

Es existieren Technische und Organisatorische Abhängigkeiten. Diese haben sowohl ein Risiko als auch einen Impact wenn das Risiko eintrifft. Dies wären folgende:

| Objekt | Abhängigkeit | Beschreibung | Status | Risiko | Impact |
|---|---|---|---|---|---|
| Foreman | VMs | Das Lifecycle Management und Provisioning System muss zur Verfügung stehen um in der Evaluationsphase Develop-VMs und in der Installationsphase Test-VMs erstellen zu können. | Im Moment ist Foreman in einer Proof of Concept Phase. | Das Risiko besteht, dass Foreman nicht betriebsbereit ist | VMs müssen von Hand aufgesetzt werden. Entsprechend wird sehr viel mehr Zeit in der Evaluations- und Installationsphase benötigt. |
| Storage | Speicher für VMs / Daten | Es müssen genügend Kapazitäten auf dem Storage vorhanden sein, um die VMs und Datenbanken in Betrieb zu nehmen | Storage wurde bereits erweitert, neue Disks für den SAN Storage wurden bestellt. | Auf dem SAN ist keine Kapazität mehr vorhanden | Es können keine VMs oder Datenbanken erstellt werden |
| Log Management / SIEM System | Sichern der Logfiles für Log Rotation | Ein Log Management System / SIEM muss vorhanden sein, um Logs langfristig sichern zu können. | Die bestehenden Plattformen für das Log Management und das SIAM werden abgelöst. Die Ausschreibung ist erfolgt | Die neue Log Management Plattform ist noch nicht betriebsbereit | Log Retention muss stark erhöht werden. Dies wird mehr Storage in Anspruch nehmen. |
| HP-UX Ablöseprojekt | Ressourcen | Das Projekt zur Ablösung der HP-UX Plattform für die Oracle Datenbanken geht in die Konzeptions- und Umsetzungsphase. | Das Projekt geht in die Konzeptions- und Umsetzungsphase. | Als Oracle DBA bin ich stark in das Projekt eingebunden. Es besteht, dass Risiko eines Ressourcenengpasses | Projekt kann nicht Zeitgemäss abgeschlossen werden |
| GitLab | Sicherung | Sicherung von Konfigurationen, Scripts usw. | GitLab ist Implementiert und Betriebsbereit. | GitLab steht nicht mehr zur Verfügung | Keine Versionierung und Teils Sicherungen mehr von Konfigurationsfiles, Scripts usw. |

Tabelle 1.8: Abhängigkeiten

## 1.4 Risikomanagement

Zusätzlich wurde eine SWOT-Analyse-Analyse für das Projekt erstellt, um weitere Risiken und Gefahren für das Projekt aufzudecken. Dabei bezieht sich die Externe Betrachtung auf die Umsysteme und die ICT des KSGR und die Interne Betrachtung auf mich und das Team um mich herum.


Abbildung 1.8: SWOT-Analyse Projekt

Aus den Abhängigkeiten und der SWOT-Analyse-Analyse wurden folgende Risiken identifiziert:

| Identifikation | | | | Abschätzung | | Behandlung | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Massnahmen ergreifen? | Zielwert | | |
| ID | Risiko | Beschreibung / Ursache | Auswirkung | WS | SM | | WS | SM | Massnahme |
| 1 | Fehlende Ressourcen | Viele parallele Projekte, Aufträge und der Tagesbetrieb | Ressourcen während der Diplomarbeit sind knapp bemessen | 3 | 4 | Ja | 2 | 2 | Organisation und Selbstmanagement |
| 2 | HP-UX Ablöseprojekt | Das Projekt ist sehr Umfangreich und ist in die Konzeptions- und Umsetzungsphase gestartet | Das Projekt wird parallel zur Diplomarbeit sehr viele Ressourcen und Aufmerksamkeit binden | 4 | 4 | Ja | 3 | 3 | Ressourcen reservieren |
| 3 | Alte Infrastruktur kann ungeplant sämtliche Ressourcen binden | HP-UX Plattform, DELL NetWorker / Data Domain Umgebung und HPE 3PAR SAN Storage Umgebung sind über dem Lifecycle und haben in den vergangenen Monaten immer wieder kritische Ausfälle erlebt | Bei einem Event, ausgelöst durch das Alter der HP-UX Plattform, der DELL NetWorker / Data Domain Umgebung oder dem SAN Storage, kann der ganze Betrieb zum erliegen kommen und entsprechend viele Ressourcen aufgrund der kritikalität binden | 4 | 4 | Ja | 3 | 3 | Monitoring vorgängig ausbauen und Massnahmen definieren |
| 4 | Schwächen beim Selbstmanagement und in der Selbstorganisation | Selbstmanagement und Organisation ist nicht meine Stärke | Das Projekt verzettelt sich, Zeit geht verloren. Auch eine folge könnte der Scope Verlust sein | 3 | 3 | Ja | 2 | 2 | Werkzeuge im Vorfeld definieren und bereitstellen |
| 5 | Scope verlust während des Projekts | Der Scope kann während des Projekts verloren gehen | Verzettelung und Zeitverlust bis hin zu scheitern | 3 | 4 | Ja | 2 | 3 | Ziele klar definieren |
| 6 | Scope Creep | Der Umfang kann stark steigen wenn Ziele nicht genau genug definiert wurden | Zeitverlust bis hin zu scheitern des Projekts | 3 | 4 | Ja | 3 | 3 | Ziele SMART definieren |
| 7 | SIEM / Log Plattform nicht betriebsbereit | Die öffentliche Ausschreibung für die neue / Log Plattform wurde erst am 23.10.2023 veröffentlicht. Bis zur Implementation kann noch Zeit vergehen. | Logs müssen länger auf dem System selber vorgehalten werden. Zudem müssen ggf. eigene Massnahmen zum Auslesen von Logs getroffen werden | 4 | 1 | Nein | | | |
| 8 | Foreman nicht betriebsbereit | Die Foreman Provisioning- und Lifecycle Plattform befindet sich aktuell erst in der Proof of Concept Phase. Dadurch besteht das Risiko, dass sie nicht betriebsbereit zum Start der Diplomarbeit ist | Ms müssen von Hand provisioniert werden. Dies bedeutet einen massiven Mehraufwand und verzögert ggf. die Evaluationsphase und mit sicherheit die Installationsphase | 3 | 5 | Ja | 3 | 4 | Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten. |

Tabelle 1.9: Risiko-Matrix der Diplomarbeit

Daraus ergibt sich folgende Risikomatrix

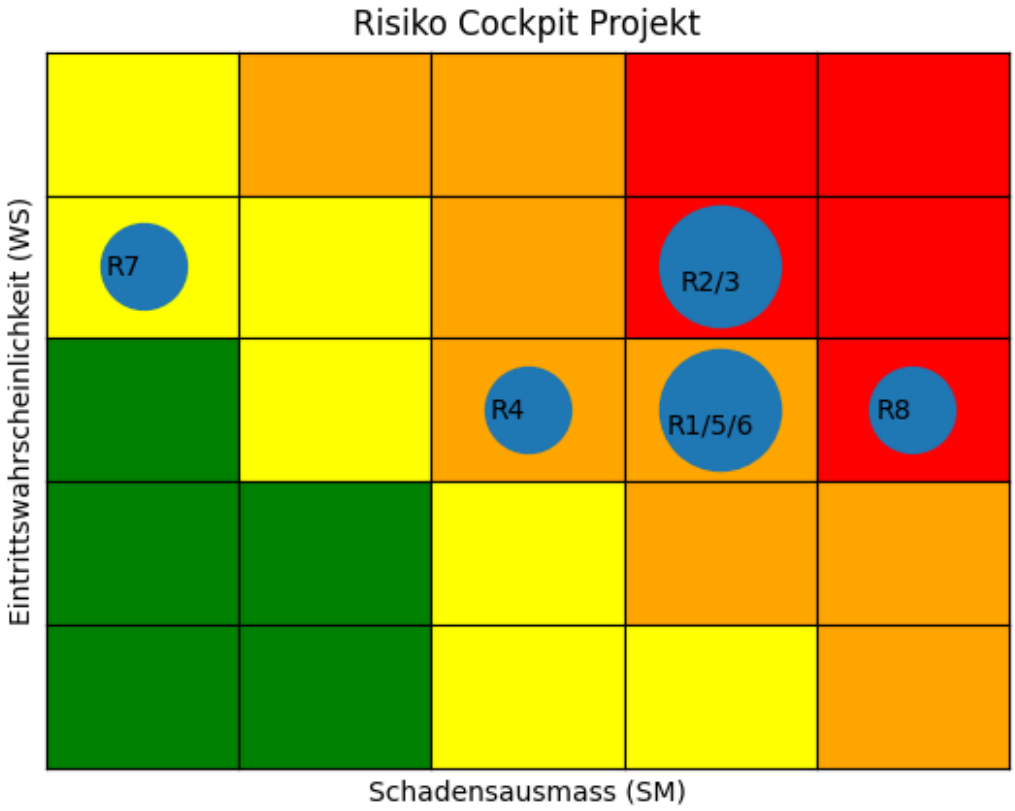


Abbildung 1.9: Projektrisiken

Mit den entsprechenden Massnahmen können die Risiken gesenkt werden:

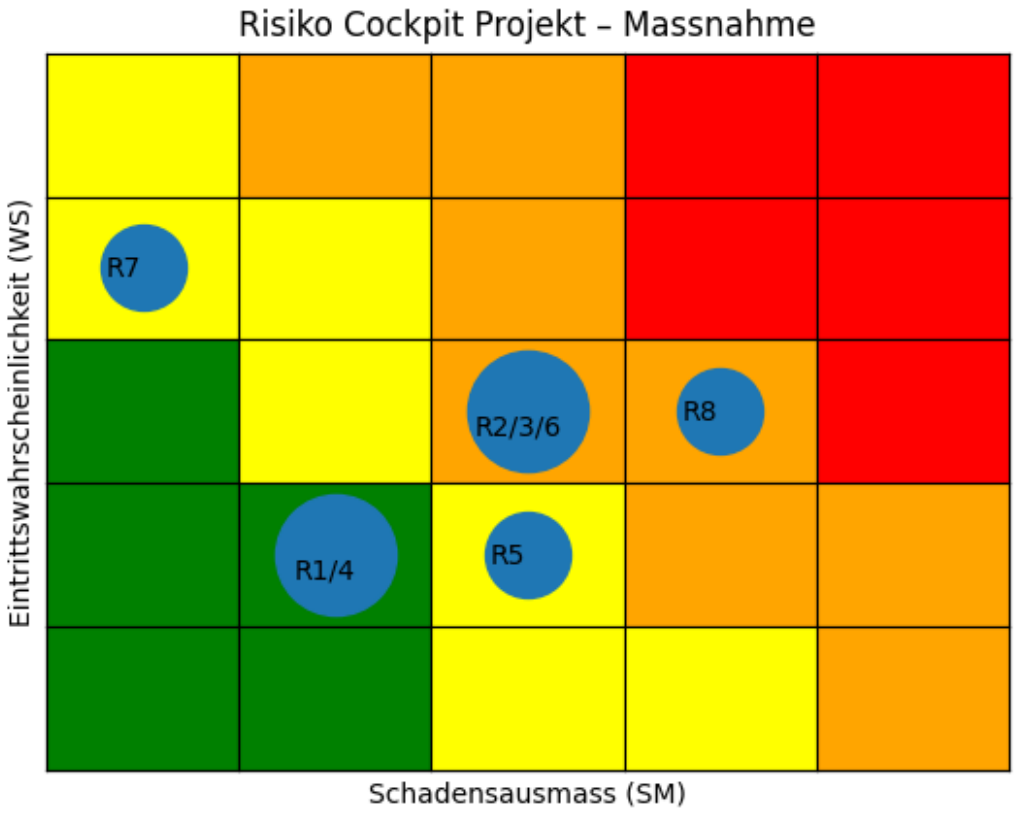


Abbildung 1.10: Projektrisiken mit Massnahmen

## 1.5 Vorgehensweise und Methoden

## 1.6 Projektmanagement

### 1.6.1 Rapport

# 2 Umsetzung

## 2.1 Evaluation

### 2.1.1 Erheben und Gewichten der Anforderungen

#### 2.1.1.1 Anforderungen

**Kostenrechnung**

Für die Kostenberechnung des Zeitaufwands wird im KSGR intern mit $120CHF/h$ gerechnet.
Jeder Arbeitstag hat dabei $8.4h$ und pro Jahr wird mit $220Tagen$ gerechnet.

**Messung des Zeitaufwands**

Der Zeitaufwand in der Evaluationsphase kann nur mit manueller Ausführung gemessen werden, da die Automatisierung nicht in der Evaluationsphase umgesetzt werden kann.
In die Evaluation einfliessen wird aber die Schätzung, wie viel Aufwand betrieben werden muss um die wichtigsten Tasks automatisieren zu können.

Folgende Messgrössen werden gestellt:

**Quorum**

**Zeitaufwand Quorum erweitern**

Bemessen wird, wie lange man braucht um einen neuen Node dem Quorum hinzuzufügen.

**Zeitaufwand Failover und Recovery**

Bemessen wird, wie lange ein Failover und ein anschliessender Recover auf den normalen Zustand dauert.

**Failover Funktionsfähigkeit**

Misst, ob der Failover bei korrekter Konfiguration funktionsfähig ist wie er vom entsprechenden System spezifiziert wurde.

**Failover Reaktionszeit**

Gemessen und bemessen wird, wie lange es im Failoverszenario dauert, bis auf einen Standby-Node umgeschaltet wird und wie lange es dauert bis offene Connections wieder voll funktionsfähig sind.

**Recoverydauer**

Bemisst, wie lange es nach einem Failover-Szenario dauert, bis der Normalzustand Widerhergestellt werden kann.

#### 2.1.1.2 Gewichtung

### 2.1.2 Exkurs Architektur

#### 2.1.2.1 Monolithische vs. verteilte SQL Systeme

Klassische SQL-Datenbanken sind Monolithische Systeme, selbst wenn sie mittels Replikation eine Primary/Standby-Architektur aufweisen. Man kann mittels eines SQL Proxys ein gewisses Mass an Load Balancing betreiben, hat aber immer noch das Problem das es einen Primary Node gibt auf dem beschrieben wird.
Verteilte Systeme wiederum

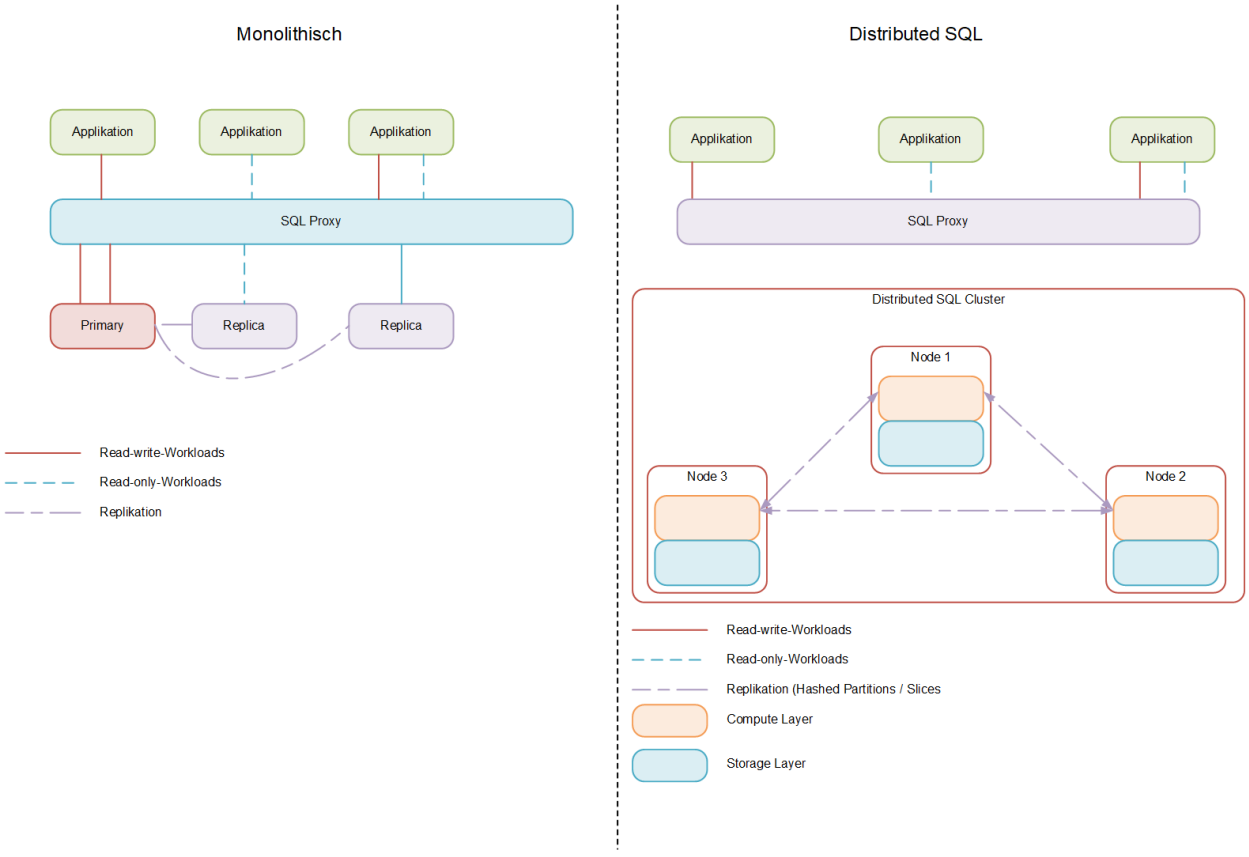


Abbildung 2.1: Monolithische vs. verteilte SQL Systeme

#### 2.1.2.2 High Availability und Replikation

Wenn eine Datenbank HA (High Availability), also Hochverfügbar, sein soll, braucht es eine Primäre und mindestens eine Sekundäre- oder Failover-Datenbank. Um Datenverlust zu vermeiden, müssen die Daten permanent von der Primären auf die sekundäre Datenbank

repliziert werden, dies nennt man Replikation[31]. Dabei wird zwischen den folgenden beiden Replikationen unterschieden:

**Synchrone Replikation**
Wenn bei einer Synchronen Replikation eine Transaktion abgesetzt wird, wird der Commit auf der primären Seite erst gesetzt, wenn die Änderung auf der sekundären Seite oder den sekundären Seiten ebenfalls eingetragen und Committed ist. Bis zu diesem Moment ist die Transaktion nicht als Committed.
Dies wird dann zum Problem, wenn keine Verbindung mehr zu mindesten einer sekundären Seite vorhanden ist. Zudem wird die Synchrone Replikation bei hohen Latenzen zum Bottleneck der Datenbank.

**Asynchrone Replikation**
Bei der Asynchronen Replikation wird eine Transaktion erst auf der eigenen primären Seite Committed und erst dann an die sekundären Nodes gesendet. Besonders bei hohen Latenzen bleibt die Datenbank immer perfomant, allerdings kann es je nach Latenz und genereller Auslastung zu Datenverlusten kommen, wenn es zum Failover kommt.

#### 2.1.2.3 Quorum

Ein Quorum-System soll die Integrität und Konsistenz in einem Datenbank-Cluster sicherstellen. Dabei gilt zu beachten, das nicht eine beliebige Anzahl an Nodes hinzugefügt werden können. Auch hat das Hinzufügen von Nodes immer eine einbusse an Performance zur Folge, besonders dann, wenn eine Synchrone Replikation gewählt wird und auf jedes Commitmend von den Replica-Nodes gewartet werden muss.

**Quorum**
: Die Mehrheit der Server, die einen funktionierenden Betrieb gewährleisten können, ohne eine Split-brainSituation zu erzeugen. Die Formel ist gemeinhin $n/2 + 1$

**Throughput**
: Beschreibt, wie sich die Anzahl Nodes auf die Schreibgeschwindigkeit der Commitments auf die restlichen Nodes auswirkt.
Die verdopplung der Server halbiert i.d.R. den Throughput.

**Fehlertoleranz**
: Beschreibt, wie viele Nodes ausfallen können, damit der Cluster noch Arbeitsfähig ist. Wobei eine erhöhung der Nodes von 3 auf 4 die Fehlertoleranz nicht erhöht da nun eine Split-brain-Situation entstehen kann.

Hier ein Beispiel wie sie in den Artikeln [29, 39, 25] beschrieben werden. Es zeigt auf, ab wie vielen Nodes die Fehlertoleranz erhöht wird und wie sich der Representative Throughput verhält.

| Anzahl Nodes | Quorum | Fehlertoleranz | Representative Throughput |
|---|---|---|---|
| 1 | 1 | 0 | 100 |
| 2 | 2 | 0 | 85 |
| 3 | 2 | 1 | 82 |
| 4 | 3 | 1 | 57 |
| 5 | 3 | 2 | 48 |
| 6 | 4 | 2 | 41 |
| 7 | 4 | 3 | 36 |

Tabelle 2.1: Quorum Beispiele

#### 2.1.2.4 CAP Theorem

Das CAP Theorem besagt, das nur zwei der drei folgenden drei Merkmale von verteilten Systeme gewährleistet werden können[16].

**Konsistenz - Consistency**
Die Datenbank ist Konsistent, alle Clients seher gleichzeitig die gleichen Daten unabhängig auf welchem Node Zugegriffen wird. Hierzu muss eine Replikation der Daten an alle Nodes stattfinden und der Commit zurückgegeben werden, also eine Synchrone Replikation stattfinden.

**Verfügbarkeit - Availability**
Jeder Client, der eine Anfrage sendet, muss auch eine Antwort erhalten. Unabhängig davon wie viele Nodes im Cluster noch aktiv ist.

**Ausfalltoleranz / Partitionstoleranz - Partition tolerance**
Der Cluster muss auch dann noch funktionsfähig bleiben, wenn es eine beliebige Anzahl von Verbindungsunterbrüchen oder anderen Netzwerkproblemen zwischen den Nodes gibt.


Abbildung 2.2: CAP-Theorem

PostgreSQL, Oracle Database oder IBM DB2 präferieren CA, also Konsistenz und Verfügbarkeit.

#### 2.1.2.5 Skalierung

Datenbanken müssen skalierbar sein. Dabei wird unterschieden zwischen einer vertikalen Skalierung (scale-up) und horizontaler Skalierung (scale-out). Bei der vertikalen Skalierung werden den DB-Servern mehr CPU-Cores und Memory sowie zum Teil Storage hinzugefügt, wobei der Storage in jedem Fall wachsen wird. Beim horizontalen Skalieren werden weitere

DB-Nodes in den Cluster eingehängt[27]:

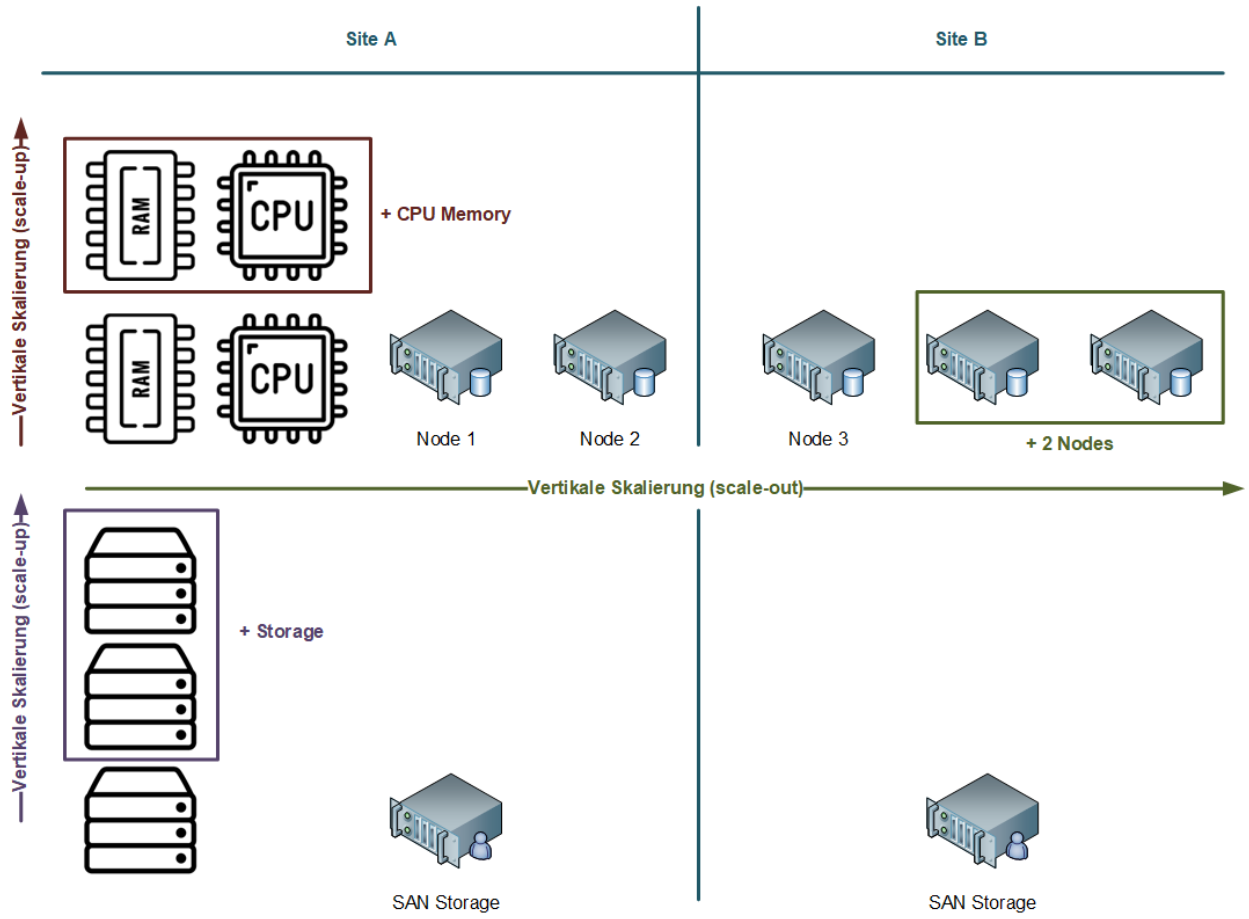


Abbildung 2.3: Datenbankskalierung

Bei monolithischen Datenbanken, werden irgendwann die grenzen der horizontalen Skalierung erreicht und man muss wieder vertikal Skalieren, um dem Primary Node genügend Rechnerleistung vorzuhalten.

### 2.1.3 Testziele erarbeiten

### 2.1.4 PostgreSQL Benchmarking

PostgreSQL bietet ein Benchmarking-Tool,[24, 1] mit dem die DB Vermessen werden kann.

### 2.1.5 Analyse gängiger PostgreSQL HA Cluster Lösungen

#### 2.1.5.1 PostgreSQL Replikation

PostgreSQL bietet von Haus aus Möglichkeiten, um Replikationen durchzuführen. Dabei ist nicht jede gleich gut für jedes Szenario geeignet[23].

**Shared Disk Failover**

**File System (Block Device) Replication**

**Write-Ahead Log Shipping**

**Logical Replication**

**Trigger-Based Primary-Standby Replication**

**Data Partitioning**
**Multiple-Server Parallel Query Execution**

#### 2.1.5.2 KSGR Lösung

Das Kantonsspital Graubünden hat basierend auf keepalived wird geprüft ob die primäre Seite erreichbar und betriebsbereit ist. Trifft dies nicht mehr zu, wird ein Failover durchgeführt[40]. Ist die primäre Seite wieder verfügbar, wird ein Restore auf die primäre Seite gefahren.
Es wird beim Restore immer ein komplettes Backup der sekundären Seite auf die primäre Seite übertragen. Ursache ist, dass die normalerweise für den Datenrestore benötigten PostgreSQL Board mittel nur für eine relativ kurze Zeit eingesetzt werden können ehe die differenzen zwischen den beiden Seiten zu gross werden.
Bei kleinen Datenbanken wie jene für Harbor und GitLab ist die Zeit die hierfür benötigt wird, nicht relevant. Sind die Datenbanken auf dem PostgreSQL Cluster jedoch grösser, kann der Restore mehrere Minuten dauern.

#### 2.1.5.3 pgpool-II

pgpool-II ist eine Middleware die zwischen einem PostgreSQL Cluster und einem PostgreSQL Client gesetzt wird. pgpool-II bietet folgende Funktionen[37, 21]:

**High Availability**
pgpool-II bietet einen automatic Failover genannten Service an, den Watchdog. Dieser schwenkt auf einen Standby-Server und entfernt den Defekten Server. Um false positive Events und Split-brains zu verhindern setzt pgpool-II auf einen eigens entwickelten Quorum-Algorithmus.

**Connection Pooling**
Bestehende Connections werden wiederverwendet um die Anzahl gleichzeitig offener Connections zu reduzieren. Der Pool wird dabei anhand von Username, Database, Protocol und weiteren Verbindungsparametern zugeordnet.

**Replikation**
Nebst dem Standard PostgreSQL bietet pgpool-II sein eigenes Replikationssystem an.

**Load Balancing**
Ähnlich wie Oracle Active Data Guard [8] bietet auch pgpool-II die Möglichkeit, SELECT-Queries und Backup-Jobs auf die Secondary-Nodes umzuleiten um den Primary Node zu entlasten.

**Limiting Exceeding Connections**
Die Anzahl an concurrent Connections, also gleichzeitiger Verbindungen, ist bei PostgreSQL begrenzt (Systemparameter wird dabei vom DBA gesetzt). pgpool-II speichert alle Connections, die über dem Limit sind, in einer Queue und somit nicht sofort fehlerhaft abgelehnt.

**Watchdog**
Der Watchdog koordiniert mehrere pgpool-II Nodes und verhindert ein Split-brain.

**In Memory Query Caching**
pgpool-II speichert SELECT-Queries in einem Cache und verwendet die ResultSets wieder, wenn eine identische Abfrage eingeht.

**Online Recovery**
pgpool-II bietet die möglichkeit, einen Online Recovery resp. eine Online Synchronisation eines Nodes durchzuführen, auch kann ein neuer Standby-Node synchronisiert werden. Dafür muss der Node aber im Detached Mode stehen, unabhängig ob der Detach manuell oder von pgpool-II ausgeführt wurde.

#### 2.1.5.4 pg_auto_failover

#### 2.1.5.5 Patroni

#### 2.1.5.6 CloudNativePG

#### 2.1.5.7 yugabyteDB - Distributed SQL 101

yugabyteDB - Distributed SQL 101 ist eine nahezu komplett PostgreSQL Kompatible Datenbank. Sie ist eine Distributed SQL Datenbank, also eine Verteilte Datenbank[49].

### 2.1.6 Installation verschiedener Lösungen

# 3 Resultate

## 3.1 Zielüberprüfung

## 3.2 Schlussfolgerung

## 3.3 Weiteres Vorgehen / offene Arbeiten

## 3.4 Persönliches Fazit

# Abbildungsverzeichnis

# Tabellenverzeichnis

## Literatur


[1] *About pgbench-tools*. https://github.com/gregs1104/pgbench-tools. original-date: 2010-02-17T13:33:28Z. 2023.

[2] Satyadeep Ashwathnarayana und Inc. Netdata. *How to monitor and fix Database bloats in PostgreSQL? | Netdata Blog*. https://blog.netdata.cloud/postgresql-database-bloat/. 2022.

[3] GitLab B.V. und GitLab Inc. *The DevSecOps Platform | GitLab*. https://about.gitlab.com/.

[4] Alexandre Cassen und Read the Docs. *Introduction — Keepalived 1.2.15 documentation*. https://keepalived.readthedocs.io/en/latest/introduction.html. 2017.

[5] Microsoft Corporation. *Azure SQL-Datenbank – ein verwalteter Clouddatenbankdienst | Microsoft Azure*. https://azure.microsoft.com/de-de/products/azure-sql/database. 2023.

[6] Microsoft Corporation. *Datenbank-Software und Datenbankanwendungen | Microsoft Access*. https://www.microsoft.com/de-de/microsoft-365/access. 2023.

[7] Microsoft Corporation. *Microsoft Data Platform | Microsoft*. https://www.microsoft.com/de-ch/sql-server.

[8] ORACLE CORPORATION. „Oracle (Active) Data Guard 19c“. In: (2019), S. 14.

[9] Varun Dhawan und data-nerd.blog. *PostgreSQL-Diagnostic-Queries – data-nerd.blog*. https://data-nerd.blog/2018/12/30/postgresql-diagnostic-queries/.

[10] Elektronik-Kompendium.de und Schnabel Schnabel. *SAN - Storage Area Network*. https://www.elektronik-kompendium.de/sites/net/0906071.htm. 2023.

[11] DB-Engines und solidIT consulting & software development gmbh. *DB-Engines Ranking*. https://db-engines.com/en/ranking.

[12] DB-Engines und solidIT consulting & software development gmbh. *relationale Datenbanken - DB-Engines Enzyklopädie*. https://db-engines.com/de/article/relationale+Datenbanken?ref=RDBMS.

[13] The Linux Foundation. *Harbor*. https://goharbor.io/. 2023.

[14] Kanton St. Gallen - Amt für Gesundheitsversorgung und Staatskanzlei Kanton St. Gallen - Dienststelle Kommunikation. *Weiterentwicklung der Strategie der St.Galler Spitalverbunde | sg.ch*. https://www.sg.ch/gesundheit-soziales/gesundheit/gesundheitsversorgung--spitaeler-s spitaeler-kliniken/spitalzukunft.html.

[15] Git. *About - Git*. https://git-scm.com/about.

[16] IBM Deutschland GmbH. *Was ist das CAP-Theorem? | IBM*. https://www.ibm.com/de-de/topics/cap-theorem. 2023.

[17] IBM Deutschland GmbH. *Was ist OLAP? | IBM*. https://www.ibm.com/de-de/topics/olap.

[18] Jedox GmbH. *Was ist OLAP? Online Analytical Processing im Überblick*. https://www.jedox.com/de/blog/was-ist-olap/. Section: Knowledge.

[19] Pure Storage Germany GmbH. *Was ist ein Storage Area Network (SAN)? | Pure Storage*. https://www.purestorage.com/de/knowledge/what-is-storage-area-network.html.

[20] Gesundheitsamt Graubünden, Uffizi da sanadad dal Grischun und Ufficio dell'igiene pubblica dei Grigioni. *Kenndaten 2016 Spitäler und Kliniken September 2018*. https://www.gr.ch/DE/institutionen/verwaltung/djsg/ga/InstitutionenGesundeitswesens/Spitaeler/Dok%20Spitler/Kenndaten%202016%20Spit%C3%A4ler.pdf.

[21] The Pgpool Global Development Group. *What is Pgpool-II?* https://www.pgpool.net/docs/44/en/html/intro-whatis.html. 2023.

[22] The PostgreSQL Global Development Group. *25.1. Routine Vacuuming*. https://www.postgresql.org/docs/16/routine-vacuuming.html. 2023.

[23] The PostgreSQL Global Development Group. *27.1. Comparison of Different Solutions*. https://www.postgresql.org/docs/16/different-replication-solutions.html. 2023.

[24] The PostgreSQL Global Development Group. *pgbench*. https://www.postgresql.org/docs/16/pgbench.html. 2023.

[25] Patrick Hunt, Mahadev Konar, Flavio P Junqueira und Benjamin Reed. „ZooKeeper: Wait-free coordination for Internet-scale systems". In: (2010).

[26] Splunk Inc. *Splunk | Der Schlüssel zu einem resilienten Unternehmen*. https://www.splunk.com/de_de. 2023.

[27] Sebastian Insausti. *Scaling PostgreSQL for Large Amounts of Data*. https://severalnines.com/blog/scaling-postgresql-large-amounts-data/. 2019.

[28] Shiv Iyer und MinervaDB. *PostgreSQL DBA Daily Checklist*. https://minervadb.xyz/postgresql-dba-d 2020.

[29] Unmesh Joshi. *Quorum*. https://martinfowler.com/articles/patterns-of-distributed-systems/quorum.html. 2020.

[30] Martin Keen und IBM Deutschland GmbH. *IBM Db2*. https://www.ibm.com/de-de/products/db2.

[31] Pasha Kostohrys. *Database replication — an overview*. https://medium.com/@pkostohrys/database-replication-an-overview-f7ade110477. 2020.

[32] Anatoli Kreyman. *Was ist eigentlich Splunk?* https://www.kreyman.de/index.php/splunk/76-was-ist-eigentlich-splunk-big-data-platform-monitoring-security.

[33] Pankaj Kushwaha und Unit 3D North Point House. *POSTGRESQL DATABASE MAINTENANCE. Routine backup of daily database... | by Pankaj kushwaha | Medium*. https://pankajconnect.medium.com/postgresql-database-maintenance-66cd638d25ab.

[34] Red Hat Limited. *Was ist CI/CD? Konzepte und CI/CD Tools im Überblick*. https://www.redhat.com/de/topics/devops/what-is-ci-cd.

[35] Switzerland Linuxfabrik GmbH Zurich. *Keepalived — Open Source Admin-Handbuch der Linuxfabrik*. https://docs.linuxfabrik.ch/software/keepalived.html. 2023.

[36] Nico Litzel, Stefan Luber und Vogel IT-Medien GmbH. *Was ist Elasticsearch?* https://www.bigdata-insider.de/was-ist-elasticsearch-a-939625/. 2020.

[37] SRA OSS LLC. *pgpool Wiki*. https://www.pgpool.net/mediawiki/index.php/Main_Page. 2023.

[38] Hewlett Packard Enterprise Development LP. *Was ist SAN-Speicher? | Glossar*. https://www.hpe.com/ch/de/what-is/san-storage.html.

[39] Diego Ongaro. „Consensus: Bridging Theory and Practice“. In: (2014).

[40] Bruno Queirós und LinkedIn Ireland Unlimited Company. *Postgresql replication with automatic failover*. https://www.linkedin.com/pulse/postgresql-replication-automatic-failover-brun C3%B3s. 2020.

[41] Kanton St. Gallen - Dienst für politische Rechte und Staatskanzlei Kanton St. Gallen - Dienststelle Kommunikation. *Wahlkreise für Kantonsratswahlen | sg.ch*. https://www.sg.ch/politik-verwaltung/abstimmungen-wahlen/wahlen/Wahlkreise-im-Kanton-SG.html.

[42] Ed Reckers und SnapLogic Inc. *Was ist die Snowflake-Datenplattform?* https://www.snaplogic.com/de/blog/snowflake-data-platform. 2023.

[43] IONOS SE. *Apache Cassandra: Verteilte Verwaltung großer Datenbanken*. https://www.ionos.de/digitalguide/hosting/hosting-technik/apache-cassandra-vorgestellt/. 2021.

[44] IONOS SE. *Datenbankmanagementsystem (DBMS) erklärt*. https://www.ionos.de/digitalguide/hosting/hosting-technik/datenbankmanagementsystem-dbms-erklaert/. 2020.

[45] IONOS SE. *MongoDB – die flexible und skalierbare NoSQL-Datenbank*. https://www.ionos.de/digitalguide/websites/web-entwicklung/mongodb-vorstellung-und-vergleich-mit-mysc 2019.

[46] IONOS SE. *SQLite: Die bekannte Programmbibliothek im Detail vorgestellt*. https://www.ionos.de/digitalguide/websites/web-entwicklung/sqlite/. 2023.

[47] IONOS SE. *Was ist Redis? Die Datenbank vorgestellt*. https://www.ionos.de/digitalguide/hosting/hosting-technik/was-ist-redis/. 2020.

[48] IONOS SE. *Was ist SIEM (Security Information and Event Management)?* https://www.ionos.de/digitalguide/server/sicherheit/was-ist-siem/. 2020.

[49] Sami Ahmed Siddiqui. *Distributed SQL 101*. https://www.yugabyte.com/distributed-sql/.

[50] Inc. Snowflake. *Datenbanken, Tabellen und Ansichten – Überblick | Snowflake Documentation*. https://docs.snowflake.com/de/guides-overview-db.

[51] Thomas-Krenn.AG. *Git Grundlagen – Thomas-Krenn-Wiki*. https://www.thomas-krenn.com/de/wiki/Git_Grundlagen.

## Glossar

**AUTOVACUUM** Der AUTOVACUUM Job räumt die Tablespaces und Data Files innerhalb von PostgreSQL sowie auf dem Filesystem nach Lösch- und Manipulations-Transaktionen auf, aktualisiert Datenbank interne Statistiken und verhindert Datenverlust von selten genutzten Datensätzen[22].. 15, 17

**Cassandra** Cassandra ist eine Spaltenorganisierte NoSQL-Datenbank die 2008 veröffentlicht[43] wurde.. 7

**CI/CD** Continuous Integration/Continuous Delivery bedeutet, das Anpassungen kontinuirlich in die Entwicklungsumgebungen integriert und auf die Zielplattformen verteilt werden[34].. 4

**DBMS** Ein Database Management System regelt und organisiert die Datenbasis einer Datenbank[44].. 4

**Elasticsearch** Elasticsearch ist eine 2010 veröffentlichte Open-Source Suchmaschine die auf Basis von JSON-Dokumenten und einer NoSQL-Datenbank arbeitet[36].. 7

**Failover** In einem Fehlerfall wird in einem HA-System meist ein Primary Node auf den Secondary ungeplant geswitched.. 17, 25, 26, 30, 41

**Foreman** Foreman ist ein Lifecycle Management und Provisioning System für Virtuelle und Physische Server. Ab Version 6 basierte der Red Hat Satellite auf Foreman. 19, 21, 42

**Git** Git ist eine Versionierungssoftware und bietet die Möglichkeit, Repositories erstellen zu können. Die Repositories sind dabei nicht zentral sondern dezentral organsiert und arbeiten daher mit Working Copies von Repositories[15, 51].. 40

**GitLab** GitLab ist ein Git basierendes System für die Versionierung und bietet dabei auch noch Dienste für CI/CD. GitLab kann sowohl als Online Dienst als auch als On-premises Service konsumiert werden[3].. 17, 19, 30

**Harbor** Harbor ist ein Open-Source-Tool zur Registrierung von Richtlinien rollenbasierten Zugriffssteuerung[13]. Harbor wird beim KSGR zur Verwaltung der Kubernetes-Plattform verwendet.. 17, 30

**HP-UX** Dieses UNIX-Derivat ist ein abkömmling von System III, System V R3 und System V R4 und wurde von HP zum ersten Mal 1982 veröffentlicht.. 4, 5, 8, 19, 21

**IBM DB2** IBM DB2 ist eine Relationale Datenbank[30] deren Vorläufer System-R von IBM zwischen 1975 und 1979 entwickelt wurde. DB2 selber wurde 1983 von IBM veröffentlicht.. 7, 28

**keepalived** keepalived nutzt VRRP um eine leichtgewichtige Lösung für ein HA-Failover zu realsieren. keepalived benötigt dazu keinen dritten Node, also einen Quorum-Node. Wenn die definierte sekundärseite keine Antwort mehr von der primären Seite nach einer definierten Anzahl versuchen in einem bestimmten Interval mehr bekommt, oder ein per Skript definiertes Event auf der primären Seite eintrifft, wird ein Failover auf die sekundäre Seite ausgeführt. Je nach Konfiguration kann der Restore auf die primäre Seite eingeleitet werden wenn diese wieder verfügbar ist oder der Restore unterbunden werden[35, 4].. 30

**Kubernetes** Kubernetes, oder k8s, ist eine Open-Source Containerplattform die ursprünglich von Google 2014 für die Bereitstellung und Orchestrierung von Containern entwickelt wurde aber 2015 an eine Tochter Foundation der Linux Foundation gespendet. Kubernetes kommt aus dem Griechischen und bedeutet Steuermann.. 4, 8, 40

**Linux** Linux ist ein Open-Source Betriebssystem, welches von Linus Torvalds 1991 in seiner frühesten Form entwickelt wurde und lose vom UNIX Derivat MINIX inspiert war. Linux besteht heute aus einer enorm grossen Anzahl an Distributionen und läuft auf einer grossen Anzahl von Plattformen.. 5, 8, 41

**MariaDB** MariaDB ist ein MySQL Fork des ehemaligen MySQL Mitbegründers Michael Widenius, wobei sich der Name Maria aus dem VOrnamen einer seiner Töchter ableitet. NAch dem Fork 2009 blieb MariaDB für eine Zeitlang sehr ähnlich mit MySQL und behielt ein ähnliches Versionierungsschema bei. Dies änderte sich 2012 wo dann direkt mit der Version 10 weitergefahren wurde. Beide Datenbanken entfernen sich im Lauf eder Zeit immer mehr voneinander undf sind nicht mehr in jdem Fall kompatibel oder beliebig austauschbar. Auf den Linux Distributionen tratt MariaDB die Nachfolge von MySQL als Standard Datenbank an.. 5, 7, 8, 9

**Microsoft Azure SQL Database** Microsoft Azure SQL Database oder auch Azure SQL ist eine Relationale Datenbank die von Microsoft für die Azure Cloud optimiert 2010 Entwickelt wurde[5].. 7

**Microsoft Access** Access wurde 1992 veröffentlicht und ist Entwicklungsumgebung, Front- und Backend-Software und Relationale Datenbank in einem[6].. 7

**Microsoft SQL Server** MS SQL Server ist das RDBMS von Microsoft[7]. Nebst Microsoft Windows und Windows Server lässt es sich seit Version 2014 ebenfalls auf Linux Betreiben. In der Wirtschaft ist die primäre Plattform aber Windows Server.. 5, 7, 42

**MongoDB** MongoDB ist eine dokumentenorientierte NoSQL-Datenbank, die zurm ersten Mal 2007 veröffentlicht wurde[45].. 7, 8

**MySQL** Die Datenbank MySQL wurde Ursprünglich als reine Relationale Open-Source Datenbank von Firma MySQL AB 1994 Entwickelt. Der Name My leitet sich vom Namen My der Tochter des Mitbegründers Michael Widenius ab. Als Sun Microsystem 2008 MySQL übernahm, hielt sich die Option frei, bei einem Kauf von Sun Microsyszem durch Oracle gründen zu dürfen. Seit Oracle Sun Microsystem 2010 gekauft hat, wurden immer mehr Funktionalitäten von der Community Edition zu der Enterprise Edition verschoben worden. Aus diesem Grund hat heute der MySQL Fork MariaDB MySQL mehrheitlich aus allen Linux Distributionen als Standard Datenbank verdrängt.. 5, 7, 8, 9

**NoSQL** NoSQL steht für Not only SQL. Das heisst, Relationale Datenbanken haben komponenten wie Dokumentendatenbanken, Graphendatenbanken, Key-Value-Datenbanken und Spaltenorientiert Datenbanken. Viele der grossen Datenbanklösungen wie Oracle Database oder Microsoft SQL Server sind NoSQL Datenbanken resp. bieten diese option an.. 7, 40, 42, 43, 44

**OLAP** Eine Online Analytical Processing, kurz OLAP, ist eine Multirelationale resp. Multidimensionale Datenbanklösung. Sie wird oft in Form eines Datenwürfels erklärt, kann aber auf verschiedene Arten umgesetzt werden[18, 17]. OLAP-Systeme bieten eine Hochperformante Analyse grosser Datenmengen und sind oftmals zentraler Teil eines Data-Warehouses.. 4, 7

**openSUSE** openSUSE ist eine Linux Distribution die aus den Distributionen Slackware Linux, jurix und S.u.S.E. Linux 4.2 heraus entstand. Die Distirbution wird heute von der SUSE S.A. betreut. Das OS ist Open-Source wobei es Enterprise Editionen für Server und Desktop sowie Tools für das Lifecycle Management und Provisioning gibt, die in Konkurenz zu Red Hat Satellite und deren Open-Source Basis Foreman stehen.. 46

**Oracle Database** Die erste verfügbare Version der Oracle Datenbank kam im Jahr 1979 mit Version 2 (statt Version 1) heraus, damals allerdings nur mit den Basisfunktionen. Im Laufe der Zeit wuchs der Funktionsumfang sehr stark an, die Grundlage des Client-Server-Designs kam erstmals im Jahr 1985 mit Version auf den Markt und hat sich im Prinzip bis heute gehalten. Mit der mit Version 8/8i 1997 erschienen Optimizer und mit der Version 9i 2001 erschienenn Flashback-Funktionalität (die ein schnelles Online Recovery sowie einen Blick in die Vergangheit ermöglichen) konnte Oracle sich stark von der Konkurenz absetzen. Heute gilt die Datenbank als erste Wahl, wenn es um Hochverfügbare Systeme, hohe Perforamce oder grosse Datenmengen geht.. 5, 7, 8, 28, 42

**PostgreSQL** Die OpenSource Datenbank PostgreSQL wurde in Form von POSTGRES zum ersten Mal 1986 von der University of California at Berkeley veröffentlicht. und zählt zu den

beliebstesten OpenSource Datenbanken. Zudem besteht in vielen bereichen eine gewisse ähnlichkeit zu Oracles Oracle Database.. 5, 7, 8, 9, 12, 13, 28, 29, 30, 31, 46

**PostgreSQL HA Cluster** Der HA Cluster des PostgreSQL Clusters. 17

**PostgreSQL Cluster** Ein PostgreSQL Cluster entspricht einer Instanz bei MS SQL oder einer Container Database wei Oracle.. 16, 17, 30, 43, 46

**PRTG** Das Monitoring System Paessler Router Traffic Grapher der Firma Paessler wurde 2003 zum erstmals veröffentlicht und war ebenfalls als Netzwerkmonitoring System konzipiert. Wie bei Zabbix lässt sich heute damit ebenfalls fast jedes IT-System damit Überwachen. Reichen die Zahlreich vorhanden Standard Sensoren nicht, können eigene Sensoren geschrieben werden. PRTG ist nicht Open-Source, man bezahlt anhand gewisser Sensor Packages.. 4, 5, 15

**Quorum** In verteilten Systemen resp. Cluster muss sichergestellt werden, das bei einem Ausfall oder ein Netzwerktrennung zwischen den Nodes es zu keiner Split-brain-Situation kommt. Hierzu wird i.d.R. ein Quorum verwendet. I.d.R. wird jener Teil des Quorums zum Primary oder alleinigen Node, der mit der die Mehrheit aller Nodes vereint. Daraus ergeben sich bestimmte grössen, mit 5 Nodes braucht es 3 Nodes um aktiv zu bleiben und mit 3 Nodes deren 2. Bei diesen Konstelationen wird daher darauf geachtet, eine ungerade Anzahl Nodes im Cluster zu halten um keine Pat-Situation zu provozieren. Im Kapitel Unterunterabschnitt 2.1.2.3 wird genauer auf die Mechanik eines Quorums eingegangen. . 30, 41

**RDBMS** Ein RDBMS ist ein Datenbankmanagementsystem für eine Relationale Datenbank. Relationale Datenbanken sind Tabellenorgansierte Datenmodelle die auf Relationen aufbauen, deren Schematas sich Normalisieren lassen. Dabei müssen Relationale Datenbanken müssen dabei auch Mengenoperationen, Selektion, Projektion und Joins erfüllen um als Relationale Datenbanken zu gelten[12].. 4, 8

**Redis** Redis ist eine Key-Value-orientierte NoSQL In-Memory-Datenbank, dh. die Daten liegen Primär im Memory und nicht auf dem Storage[47]. Redis wurde 2009 zum ersten Mal veröffentlicht.. 7, 8

**SAN** Ein Storage Area Network ist ein dediziertes Netzwerk aus Storage Komponenten. SAN Systeme bieten redundante Pools an Speicher. Die Physischen Festplatten werden zu Virtuellen Lunes, also logischen Einheiten, zusammengefasst. Dies werden nach aussen den Konsumenten präsentiert[10, 38, 19]. 4, 5, 19, 21

**SIEM** Ein sammelt Daten aus verschieden Netzwerkkomponenten oder Geräten von Agents oder Logs. Diese Daten werden permanent analysiert und mit einem definierten Regelwerk gegengeprüft. Ziel ist es, verdächtige Events zu erkennen und einem Angriff zuvorzukommen oder ihn möglichst früh zu unterbinden[48].. 4, 19

**Snowflake** Snowflake ist eine Big Data Plattform die Data Warehousing, Data Lakes, Data Engineering und Data Science in einem Service vereint. Die Daten werden in eigenen internen Relationalen und NoSQL-Datenbanken gespeichert[50, 42]. 7

**Split-brain** Im Kapitel **??** werden die ursachen und folgenden eines Split-brains genauer besprochen. . 26, 43

**Splunk** Splunk ist Big Data Plattform, Monitoring- und Security-Tool in einem[26, 32]. . 7

**SQLite** SQLite ist eine Relationale Embedded Datenbank welche seit 2000 existiert. Sie verzichtet auf eine Client-Server-Architektur und kann in vielen Frameworks eingebunden werden[46].. 7

**Switchover** In einem Maintenance-Fall in einem HA-System meist ein Primary Node auf den Secondary geplant geswitched.. 17

**SWOT-Analyse** Eine SWOT-Analyse soll die Stärken (Strengths), Schwächen (Weaknesses), Chancen (Opportunities) und Risiken (Threads) für ein Unternehmen oder ein Projekt aufzueigen. Anhand einer SWOT-Analyse werden i.d.R. anschliessend Strategien abgeleitet um mit den Stärken und Chancen die Schwächen und Risiken abzufangen oder anzumildern.. 4, 20

**UNIX** Die erste Version von UNIX wurde im Jahr 1969 in den Bell Labs entwickelt und übernahm viele Komponenten aus dem gescheiterten Multics-Projekt. Aus dem Ursprünglichen UNIX enstanden im Laufe der Zeit viele offene und Proprioritäre Derivate deren Einfluss weit über die Welt der Informatik reicht.. 4

**VRRP** VRRP . 4, 41

**Zabbix** Das 2001 veröffentlichte Open-Source Monitoring System Zabbix gilt zwar als Netzwerk-Monitoring System, allerdings kann heute nahezu jedes IT-System damit überwacht werden. Zabbrix speichert die Metriken und nicht die Auswertungen, das heisst, solange die Daten vorhanden sind können Grafiken zu jedem Zeitpunkt generiert worden. Zabbix ist grundsätzlich Open-Source, man kann allerdings Supportverträge Abschliessen.. 9

## Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich diese Thesis selbständig verfasst und keine andern als die angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen wurden, habe ich als solche kenntlich gemacht. Ich versichere zudem, dass ich bisher noch keine wissenschaftliche Arbeit mit gleichem oder ähnlichem Inhalt an der Fernfachhochschule Schweiz oder an einer anderen Hochschule eingereicht habe. Mir ist bekannt, dass andernfalls die Fernfachhochschule Schweiz zum Entzug des aufgrund dieser Thesis verliehenen Titels berechtigt ist.

---

Ort, Datum, Unterschrift

#### 0.0.1 Rapport

#### 0.0.2 pgpool-II

##### 0.0.2.1 PostgreSQL Cluster Installation

**PostgreSQL Package Repository in Debian einbinden**

```
linux-4fi4:~ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    group default qlen 1000
    link/ether 00:50:56:33:d8:e4 brd ff:ff:ff:ff:ff:ff
    inet 172.23.150.140/20 brd 172.23.159.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe33:d8e4/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    group default qlen 1000
    link/ether 00:0c:29:e0:2f:ec brd ff:ff:ff:ff:ff:ff
    inet 192.168.210.1/24 brd 192.168.210.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fee0:2fec/64 scope link
       valid_lft forever preferred_lft forever
```

Listing 1: openSUSE - Netzwerk Settings