UNIVERSITÄT LEIPZIG
Fakultät für Mathematik und Informatik
Institut für Informatik

Database-as-a-Service: Übersicht

im Rahmen des Seminars Cloud Data Management WS09/10

Seminararbeit

Leipzig, 17. März 2010
Betreuer: Sabine Maßmann

vorgelegt von

Christian Kötteritzsch
geb. am: 19.03.1986

Studiengang Master(neu) Informatik

Inhaltsverzeichnis

1 Einleitung

Heutzutage spielen Datenbanken eine wichtige Rolle in der Informationstechnologie. Da jedoch der Aufwand ein konventionelles Datenbankmanagementsystem zu betreiben sehr hoch ist, geht der Trend zu *Datenbanksystemen als Dienstleistung*. Dadurch ist es nicht mehr nötig eine Datenbank lokal zu installieren und zu verwalten und es fallen keine Kosten für die Administration und den Betrieb an. Der Kunde zahlt nur für die Leistungen die er wirklich benötigt und erhält eine gut skalierbare Anwendung. Der Anbieter dieser Dienstleistungen erzielt somit eine bessere Auslastung seiner Server und kann dadurch die Kosten für den Betrieb der Server reduzieren. Ein weiterer Vorteil von Datenbanken als Service ist, dass es Umweltschonender ist, da nicht jedes Unternehmen einen eigenen Server betreiben muss.

Um Datenbanksysteme als Dienstleistungen anbieten zu können, wird in Anlehnung an die Prinzipien der serviceorientierten Architektur nur noch abstrakte Services bereitgestellt, die die Datenverwaltungsfunktionalität bereitstellen. Es ist also nur die Schnittstellenbeschreibung bekannt, aber die Implementierungsdetails und die verwendete Software bleiben verborgen. Eine Beispielanwendung, die diese Prinzipien umsetzt, ist der *Amazon Simple Storage Service (S3)*, der zur Speicherung schlüsselindexierter BLOBs dient. Ebenfalls zu nennen ist die *Amazon SimpleDB*, die zur Speicherung untypisierter Multi-Maps verwendet wird. Einige weitere Anbieter von Cloud Computing Lösungen werden in Abbildung 1 dargestellt.

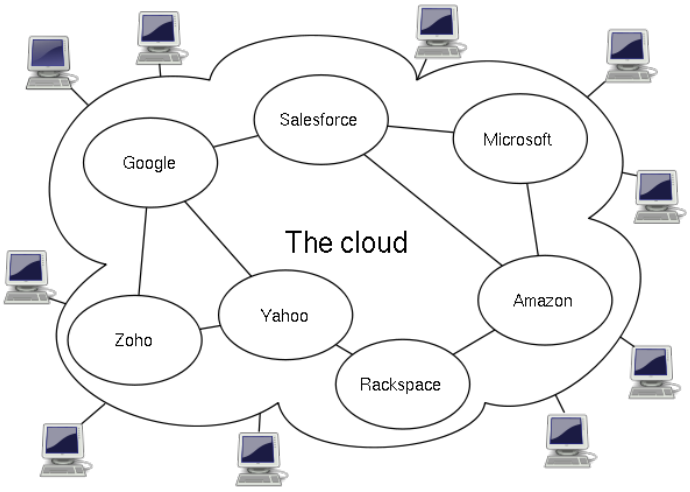

Abbildung 1: Anbieter von Cloud Computing Dienstleistungen.

Trotz dieser vielen Vorteile, die Datenbanken als Service bringen, gibt es auch einige Probleme die es noch zu beseitigen gibt, bevor solche Services für alle zur Verfügung gestellt werden können. Ein wichtiges Forschungsgebiet ist dabei die *Gewährleistung von Datenvertraulichkeit, Vollständigkeit*

der Daten und die Korrektheit der Daten.[Lan]

Diese Arbeit soll einen Überblick über Database-as-a-Service geben und beschäftigt sich in Kapitel 2 mit Software-as-a-Service und Cloud Computing. Dabei gibt es generelle Auskunft über dieses Thema und zeigt Vor - und Nachteile dieser auf. Danach wird im Kapitel 4 die Grundlagen von Database-as-a-Service behandelt, welche dann im Kapitel 5 durch ein Modell und Beispielimplementierungen vertiefend dargestellt werden. Im letzten Kapitel wird noch einmal alles zusammengefasst und bewertet.

2 Grundlagen

In diesem Kapitel werden die Grundlagen, wie *Software-as-a-Service* und *Cloud Computing* genauer vorgestellt und ihre Vor - und Nachteile aufgezeigt.

2.1 Software-as-a-Service

Dieses Kapitel gibt Auskunft über Software-as-a-Service und zeigt Vor- und Nachteile dieser Methode der Softwareentwicklung auf.

Bei Software-as-a-Service stellt ein Anbieter seine Software über Internet für andere zur Verfügung. Dabei kann der einzelne Nutzer diese Software mieten und zahlt so nur für die Zeit in der er diese nutzt. Somit erhält der Nutzer dieser Services Zugang zu neuer Software, ohne dabei jedoch die dafür benötigte Hardware, Leute die sich um die Administration kümmern und eine Lizenz dieser Software zu besitzen oder anzuschaffen. Der Serviceanbieter ist für die Einhaltung der versprochenen Leistungen des jeweiligen Services verantwortlich. Da die Software bei dem Anbieter installiert ist, kann er diese von einem Ort aus managen, ohne zum jeweiligen Kunden gehen zu müssen.

Es gibt zwei Arten wie Software-as-a-Service realisiert werden kann. Zum einen als Webanwendung, bei der die Software über einen Webserver oder Webclient genutzt wird. Bei dieser Variante spielen Webserver und Applikationsserver eine große Rolle. Zum anderen kann solch ein Service über eine Terminalanwendung genutzt werden, bei der die Desktopanwendung auf einem entfernten System installiert ist. Hier wird in der Regel ein Terminalserver und ein Thin Client für den Zugriff eingesetzt.

Bei diesem Modell gibt es Vorteile für beide Seiten, dem Nutzer und dem Anbieter dieser Services. Die Vorteile des Anbieters sind, dass er schneller Softwareveränderungen für den Kunden bereitstellen kann. Da die Software Zentral beim Anbieter verwaltet wird, kann dieser eine schnellere Anpassung für den jeweiligen Kunden vornehmen. Die Vorteile für den Kunden sind, dass er die Verantwortung für die Verwaltung der Software und der Daten des Unternehmens dem Anbieter überträgt. Er hat somit keine Kosten für die Anschaffung der nötigen Ressourcen und der Software selbst. Auch die Kosten für die Administration dieser fallen weg. Er bekommt eine nutzungsbasierte Abrechnung und zahlt somit nur das was er auch wirklich benötigt hat. Somit erhält der Kunde Zugriff auf Software, die er nicht besitzt und sich eventuell auch nicht leisten kann.

Es gibt aber auch einige Nachteile bei diesem Modell. Da die Verantwortung der Verwaltung der Software für viele Nutzer jetzt beim Anbieter liegt und diese sehr Komplex sein kann, muss sich der Anbieter eine Möglichkeit überlegen um damit fertig zu werden. Er muss auch für jeden Kunden die im Vertrag festgelegten Leistungen erfüllen können und eventuell Anpassungen an den Services vornehmen. Der Nutzer dieser Services wird sehr schnell abhängig vom Anbieter und da die Unternehmensdaten ebenfalls beim Anbieter liegen muss ein sehr gutes Vertrauensverhältnis zum Anbieter

vorliegen. Des Weiteren muss auch eine permanente Internetverbindung zum Anbieter bestehen, da sonst keine Nutzung der Services möglich wäre. Da diese Internetverbindung eine höhere Latenzzeit hat als lokal installierte Software, ist die Nutzung von Services meist langsamer.[FH00]

2.2 Cloud Computing

In diesem Kapitel wird kurz vorgestellt was Cloud Computing ist und welche Vorteile es bringt.

Cloud Computing ist ein neuer Trend dessen Ziel es ist Soft- und Hardwareressourcen wie CPU Zeit, Speicherkapazität und Netzwerkbandbreite, die nicht benötigt wird, für andere Kunden über das Internet bereit zu stellen. Dies wird in Abbildung 2 dargestellt. Dabei bezieht sich die Bereitstellung der Software auf das im vorherigen Kapitel erwähnten Software-as-a-Service und das Bereitstellen von Hardwareressourcen ist die Cloud. Der Nutzer braucht somit keine eigene Hardware anschaffen und hat keine Kosten für den Betrieb dieser, kann aber trotzdem auf Technologien zurückgreifen, die er gar nicht besitzt. Er erhält dadurch eine Lösung die gut auf die jeweils variierenden Bedürfnisse angepasst werden kann. Dadurch kann eine *nutzungsbezogene Abrechnung* erfolgen und der Kunde zahlt nur das was er an Leistungen benutzt hat. Des Weiteren braucht sich der Kunde keine Gedanken über die Verfügbarkeit, Zuverlässigkeit oder Backups seiner Daten zu machen. Diese Aufgaben übernimmt der Anbieter, wobei die genauen Spezifikationen der Leistungen die ein Anbieter erfüllen muss, in den *Service Level Agreement (SLA)* festgelegt werden. Dabei werden nicht nur die Anforderungen an den Anbieter festgelegt, sondern auch Strafen bei Vertragsverletzungen. Somit reduziert man nicht nur seine Kosten für die Anschaffung und den Betrieb der Ressourcen, sondern reduziert auch das Risiko eines überlasteten oder nicht ausgelasteten Servers.

Neben diesen Vorteilen gibt es aber auch eine Reihe von Nachteilen beim Cloud Computing. Ein Nachteil ist, das obwohl diese Services sehr mächtig sind, können sie für manche Aufgaben nicht geeignet sein. Ein weiterer Nachteil ist das die Anbieter meist nur eine lockere Datenkonsistenz anbieten. Wenn diese nicht ausreicht muss man diese direkt in die Anwendung implementieren. Außerdem ist Infrastruktur die nicht lokal verfügbar ist oft langsamer, durch die Latenzzeit des Netzwerkes[BFG+09][AFG+09].

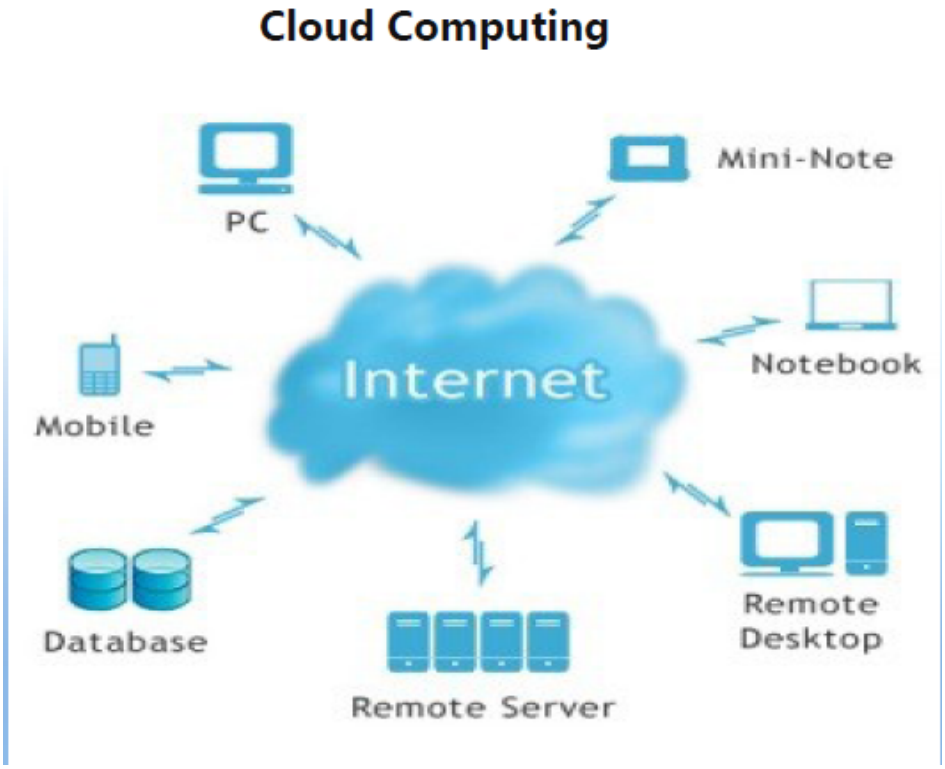

Abbildung 2: Darstellung von verschiedenen Geräten die für Cloud Computing genutzt werden. Dabei stellen die einzelnen Geräte verschiedene Hardwareressourcen bereit. Diese werden über das Internet miteinander verbunden, so dass es für den Nutzer transparent bleibt welche Ressourcen von welchem Gerät kommen. Dadurch entsteht der Eindruck der Nutzer greift nur auf ein Gerät zu.

3 Einführung in Database-as-a-Service

Da man die Prinzipien *serviceorientierter Architekturen* auch auf Datenbanksysteme anwenden kann, entsteht ein Datenbanksystem durch Komposition autonomer, lose gekoppelter Services, welche die Teilfunktionen bereitstellen. Weiterhin ist es als Weiterentwicklung möglich einzelne Services von verschiedenen Anbietern bereit stellen zu lassen. Wie man in Abbildung 3 sehen kann ist Database-as-a-Service in zwei Konzepte gegliedert, den *IT-Dienstleistungen* und den *Datenbanksystem-Services*. Die IT-Dienstleistungen bestehen im wesentlichen aus dem *vollständigen Fremdbetrieb* der erbrachten Services, die durch die Service-Level-Agreements festgelegte Güte der Dienstleistungen, die *Skalierbarkeit* der angebotenen Leistungen auf das vom Kunden benötigte Maß und die Nutzungsbasierte Abrechnung dieser Services. Datenbanksystem-Services sind entweder *vollständige Datenbanksysteme als Service* oder *Services die Teilfunktionen eines Datenbanksystemes bereitstellen*. Dabei können mehrere Teildatenbanksysteme, durch Komposition mehrerer Services, von eventuell verschiedenen Anbietern, zusammengeschalten werden, so dass ein vollständiges Datenbaksystem entsteht. Dabei müssen die so entstandenen Services im weitesten Sinne *mandantenfähig* sein, so dass jeder Kunde ein gemeinsames Datenbanksystem verwenden kann und nicht jeder eine Einzelinstanz zugewiesen wird. Beide Konzepte werden durch Services im Internet bereitgestellt bzw. über das Internet in Anspruch genommen.[Lan]

Dabei ergeben sich neue Herausforderungen um solche Datenbanksysteme betreiben zu können. Diese werden im nächsten Abschnitt genauer erläutert.

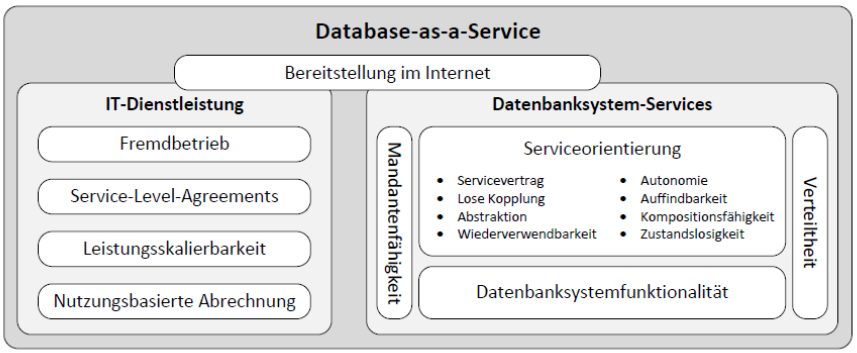

Abbildung 3: Konstituierende Merkmale von Database-as-a-Service

3.1 Herausforderungen

Um Datenbankfunktionalität über das Internet von Dritanbietern bereitstellen zu lassen zu können, müssen vorher einige neu entstanden Herausforderungen geklärt werden. In diesem Abschnitt werden einige dieser Herausforderungen vorgestellt und einige Ansätze zur Lösung dieser geboten.

Mandantenfähigkeit

Da es den Anforderungen für eine handhabbare und kosteneffiziente Infrastruktur widerspricht, jedem Kunden eine eigenes System zur Verfügung zu stellen, müssen andere Wege gefunden werden um die Mandantenfähigkeit zu realisieren. Trotzdem muss eine angemessene Isolation und Datensicherheit zwischen den einzelnen Kunden und dem Dienstleister gewährleistet sein. Die Mandantenfähigkeit ist ein schwieriger Aspekt, da man zwischen Sicherheit, Kosten und individuellen Anforderungen der Kunden abwägen muss. Dabei gibt es verschiedene Varianten die Trennung der Mandanten umzusetzen. Das Spektrum geht von völliger Isolation der Kundendaten, bis zu einem gemeinsam genutzten Schema, wo die Daten durch eine Kunden-ID unterschieden werden kann. Das Nutzen eines gemeinsamen Schemas ist nur möglich wenn ein gemeinsames Schema vorhanden ist, ansonsten kann maximal die gleiche Datenbankinstanz genutzt werden. Umso weiter unten in der Schichtenarchitektur die Isolation angesetzt wird, umso unkomplizierter ist die Umsetzung. Den Datenschutz kann man durch die Verschlüsselung der Daten realisieren und es gibt auch schon einige Ansätze, die dieses Thema inklusive der Abfrageverarbeitung behandeln.

Verwaltung der Infrastruktur

Der Kunde muss sich weder um die Administration und Wartung von Hard- und Software, noch um die Anschaffung dieser kümmern. Diese Verantwortung liegt nun gebündelt beim Anbieter, der seine Systeme optimal ausgelastet haben will, um den bestmöglichen Gewinn zu erzielen. Dazu ist es notwendig bei Bedarf neue Ressourcen hinzuzufügen oder nicht benötigte Ressourcen abzuschalten, um die mit dem Betrieb nicht benötigter Ressourcen verbundenen Kosten zu vermeiden. Dabei ist nicht davon auszugehen, dass der Anbieter dieser Dienstleistungen über eine homogene Infrastruktur verfügt. Um mit dieser steigenden Komplexität umgehen zu können, muss das System über Automatismen verfügen, die eine Echtzeitüberwachung aller Ressourcen, die automatische Stilllegung und Bereitstellung von Ressourcen, je nach Bedarf des Kunden, in einem heterogenen Umfeld zur Verfügung stellt. Diese Anforderungen werden im Autonomic Computing thematisiert.

Überwachung der SLA's

Es ist im Rahmen der Dienstleistungen nötig das System zu überwachen und die Leistung zu protokollieren, um mögliche Servicegarantien zu gewährleisten und eine Abrechnung entsprechend der Nutzung zu erstellen. Des Weiteren ist für eine Berechnung von Vertragsstrafen bei Nichteinhaltung von Garantien eine Protokollierung der Leistungen nötig. Um diese Daten erfassen zu können ist eine

Monitoring-Komponente von Nöten, die alle erforderlichen Daten sammelt, ohne jedoch die Leistung des Gesamtsystems zu beeinträchtigen. Da bereits eine Komponente zur Überwachung der Infrastruktur vorhanden sein muss, kann die Überwachung der SLA's und die Erfassung der Abrechnungsdaten von dieser mit übernommen werden.

Weitere Herausforderungen

- **Caching von Objekten**

  Da der Datenaustausch zwischen Kunde und Dienstleister hohe Kommunikationskosten zur Folge hat, gilt es diese durch die Verwendung von bestehenden *Caching-Verfahren* aus den Client/Server Datenbanksystemen zu verringern.

- **Zugriffskontrolle**

  Da die Zugriffskontrolle in der 5. Schicht angesiedelt ist, aber die vorgestellten Dienstleistungen aus weniger Schichten bestehen, muss die Zugriffskontrolle auf einer tieferen Schicht umgesetzt werden, da diese zum Umfang eines jeden Daten-Management-Dienstes gehört.

- **Administration durch den Kunden**

  Kundenseitige Änderungenswünsche an den Dienstparametern sind ein realistisches Szenario. Da diese neben den administrativen Tätigkeiten auch Anpassungen an den für den Kunden bereitgestellten Ressourcen und dem Abrechungsmodell zur Folge hat, ist zu untersuchen, ob diese durch Automatismen durch den Dienstleister unterstützt werden können.

- **Portierung von Anwendungen**

  Da Database-as-a-Service auch als alternative Backend-Lösung für bestehende Anwendungen attraktiv sein sollte, muss der Dienstleister möglichst alle *SQL-Dialekte* beherrschen. Um eine Anwendungsportierung für den Kunden möglich zu machen ist es erforderlich, dass der Dienstleister Unterstützung und Konzepte für die Portierung der Daten bereitstellen. Dabei kann der Dienstleister auf Mappingsprachen zurückgreifen um dies zu realisieren.

- **Schaffen von Vertrauen**

  Fehleranalyse, Debugging und Reporting sind nur eingeschränkt möglich, weil die Daten des Kunden beim Dienstleister liegen. Deshalb sind Exportfunktionen, Monitoring-Werkzeuge und Analysefunktionen zur Datenqualität nötig. Damit erhält der Kunde Kontrolle über seine Inhalte und die Möglichkeit diese selbst zu verwalten. Des Weiteren kann mit Hilfe von Daten-Zertifizierung das Vertrauen gestärkt werden.[KK]

3.2 Vergleich zwischen Database-as-a-Service und lokal installierter Datenbank

In diesem Abschnitt werden lokal installierte Datenbanken mit Database-as-a-Service verglichen und mögliche Vor- und Nachteile beider dargestellt (Tabelle 1).

Vorteile Database-as-a-Service

Ein wesentlicher Vorteil von Database-as-a-Service ist, dass keine Installation von Datenbanksoftware nötig ist, denn diese wird von den Anbietern bereitgestellt. Daraus ergeben sich weitere Vorteile, da keine Kosten für die Anschaffung und den Betrieb von Hard- und Software nötig ist. Ebenfalls entstehen keine Kosten in Form von administrativen Aufwand, da auch dafür der Anbieter der Dienstleistung zuständig ist. Da keine eigenen Server notwendig sind, sondern die Server der Anbieter besser ausgelastet sind, reduziert dies die Stromkosten und es findet eine umweltschonendere Nutzung der Ressourcen die zur Verfügung stehen statt. Bei Database-as-a-Service muss sich der Nutzer auch nicht um die Skalierbarkeit kümmern, da der Anbieter je nach Bedarf Ressourcen dazuschalten oder abschalten kann. Dadurch entsteht der Eindruck, der Nutzer habe eine unbegrenzte Speicherkapazität zur Verfügung. Der Betrieb eines Datenbankservers, der 24 Stunden und 7 Tage die Woche läuft, ist eine sehr kostenintensive Aufgabe für das jeweilige Unternehmen. Da einige Unternehmen sich dies nicht Leisten können, bietet sich Database-as-a-Service als Lösung an, da der Anbieter (je nach in den SLA's festgelegten Vertragsbedingungen) einen Betrieb rund um die Uhr gewährleisten muss. Ein weiterer Vorteil ist, dass im Verlauf des Tages unterschiedliche Leistungsprofile erreicht werden. Bei einer Eigenlösung müsste auch für den nichtausgelasteten Server bezahlt werden, aber bei Database-as-a-Service wird nur für die tatsächlich benötigte Leistung bezahlt. Das hat auch zur Folge, dass der Anbieter seine Server besser auslasten kann, da er je nach Last Ressourcen dazu- oder abschalten kann. Auch um den Datenschutz muss sich der Anbieter kümmern, somit kann der Nutzer die gesamte Verantwortung dafür auf den Anbieter übertragen und muss sich keine Gedanken über ein Datenbackup machen muss. Somit muss der Dienstleister sicherstellen, dass bei Ausfall von Ressourcen andere vorhanden sind um die Aufgabe zu übernehmen und bei Plattenausfall müssen alle Daten wiederhergestellt werden.

Nachteile Database-as-a-Service

Bei Database-as-a-Service gibt es aber nicht nur Vorteile sondern auch Nachteile. Ein sehr gravierender Nachteil ist, dass man vollständig vom Anbieter abhängig ist und man kann z.B. keine manuellen Optimierungen vornehmen ohne vorher den Anbieter zu kontaktieren. Da die Daten vollständig beim Anbieter liegen, ist eine Anbindung des Kunden nur über Netzwerk möglich. Somit können neue

Daten nur über das Netzwerk hinzugefügt oder verändert werden. Durch die hohe Latenzzeit gegenüber lokal installierten Datenbanken ist dies aber viel langsamer als wenn die Datenbank lokal zur Verfügung steht. Ein weiterer Nachteil von Database-as-a-Service ist, dass noch nicht alle Datensicherheitsaspekte geklärt sind, wie z.B. die einzelnen Kunden nur auf ihre Daten und nicht die von anderen Kunden zugreifen können.

Vorteile lokal installierter Datenbank

Es gibt aber auch Vorteile lokal installierter Datenbanken. Zum einen hat man vollständige Kontrolle über die Datenbank und kann schneller Änderungen vornehmen. Man hat auch eine viel geringer Latenzzeit bei lokal installierten Datenbanken als bei Database-as-a-Service. Zum anderen sind die Unternehmensdaten im Unternehmen selbst und es müssen somit keine Daten an ein fremdes Unternehmen übertragen werden. Somit muss auch kein sehr gutes Vertrauensverhältnis zu einer fremden Firma vorliegen.

Nachteile lokal installierter Datenbank

Die Nachteile einer lokal installierten Datenbank sind die hohen Kosten für die Anschaffung der benötigten Hard- und Software, die Administration dieser und der hohe Stromverbrauch durch das Betreiben der benötigten Server (auch bei gering ausgelasteten Servern). Da die Daten im Unternehmen liegen muss sichergestellt werden, dass sie im Fehlerfall immer noch verfügbar sind. Dazu sind Backuplösungen von Nöten um die sich das Unternehmen kümmern muss. Um diese Lösungen realisieren zu können entstehen weitere Kosten.[Lan][ASS+09][KK]

	DaaS	lokale Installation
Kosten für Hard - und Software	+	-
Kosten für Betrieb und Administration	+	-
Skalierbarkeit	+	-
Latenzzeit	O	+
Flexibilität	+	O
Datenschutz und Datensicherheit	O	+
Vertrauen zu Fremdfirmen	-	O

Tabelle 1: Vor - und Nachteile von Database-as-a-Service (DaaS) und lokal installierter Datenbank.

4 Das Database-as-a-Service Model

Im folgenden Kapitel wird das Database-as-a-Service Model vorgestellt und anhand von zwei Beispielen die Umsetzung dieses erklärt.

Das Database-as-a-Service Model teilt seine Services in vier Gruppen ein, den *Storage Service*, den *Access Service*, den *Data Service* und den *Extension Service*. Die einzelnen Services werden im Folgenden genauer beschrieben.

Storage Service

Bei dem Storage Service handelt sich um einen Dienst der auf Byteebene arbeitet und sehr eng mit den Dateisystemfunktionen zusammenarbeitet. Dadurch werden ähnliche Funktionen wie die ersten beiden Schichten aus dem 5 Schichten Datenbanksystemmodell zur Verfügung gestellt.

Access Service

Der Access Service ist für die physische Repräsentation der Datensätze verantwortlich und es wird dadurch eine Navigation durch verschiedene logische Datensätze und die Ausführung von Operationen ermöglicht. Es setzt auf den vorangegangenen Service auf und repräsentiert die dritte und vierte Schicht des 5 Schichten Datenbanksystemmodells.

Data Service

Der Data Service ähnelt der fünften Schicht des 5 Schichten Modells und bietet Zugriff auf die logischen Datenstrukturen, wie z.B. Tabellen oder Sichten.

Extension Service Durch den Extension Service können Erweiterungen, wie z.B. neue Datentypen zur Verfügung gestellt werden. Dabei greift dieser Service auf die vorherigen Services zurück.

In diesem Abschnitt wird das Modell noch etwas verfeinert. Dieses wird in zwei Teile gegliedert, den *Basisservices*, welche in Abbildung 4 dargestellt sind und in die *erweiterten Services*, welche in Abbildung 5 dargestellt sind. Die grundlegende Funktionalität wird durch die Basisservices bereitgestellt und erweiterten Services stellen die benötigten Funktionen im Zusammenhang mit Database-as-a-Service zur Verfügung. In den folgenden Abschnitten werden die beiden Services genauer vorgestellt.

Die Basisservices

Bei den Basisservices handelt es sich um *Speicher-, Zugriffs-, und Datenservice*, welche ergänzt werden durch einen textitTransaktionsservice, der Funktionen für die Transaktionsunterstüzung bereit stellt und einen *Metadatenservice*, der Metadaten für verschiedene Datenmodellebenen und andere Metadaten wie bspw. Zugriffsrechte verwaltet.

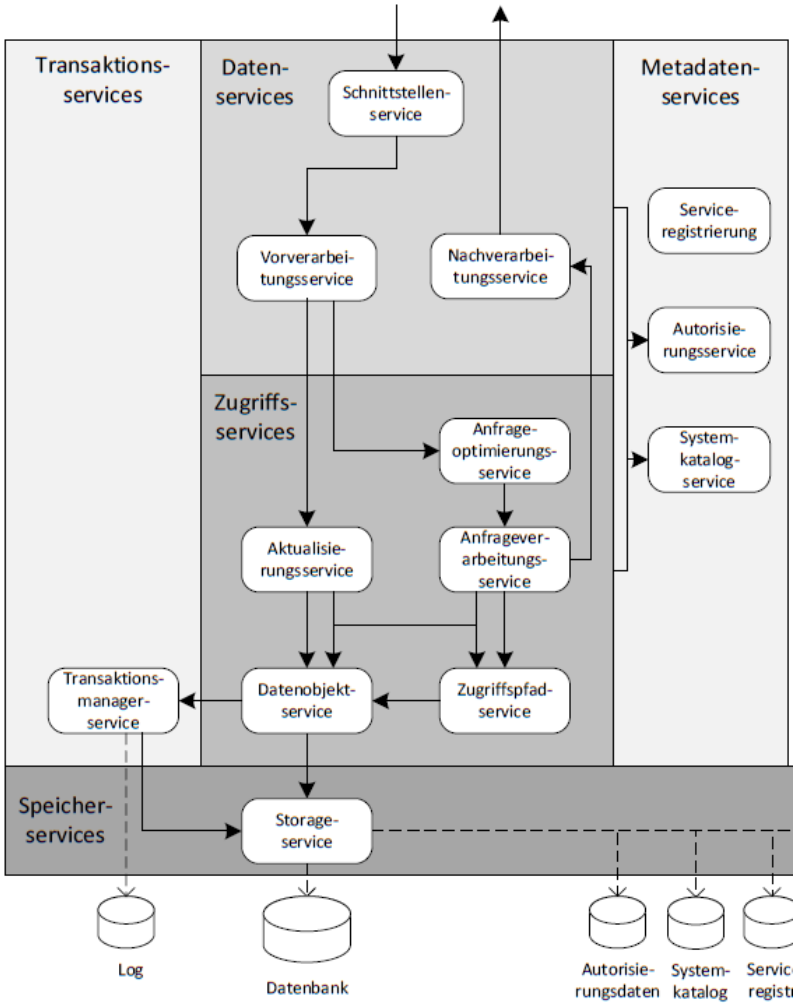

Abbildung 4: Basisservices

In Abbildung 4 sieht man die Basisservices, wobei der *Schnittstellenservices* eine Zentrale Rolle des Datenservices einnimmt. Dieser nimmt Anfragen entgegen und prüft mit Hilfe des *Autorisierungsservices* ob die Ausführung durchgeführt werden kann. Der *Vorverarbeitungssyservice* übernimmt durchzuführende Transformationen wie z.B. die Überführung eines externen Schemas in ein konzeptionelles Schema. Der *Nachverarbeitungsservice* bearbeitet Anfrageergebnisse nach. Bei lesenden Operationen werden mit Hilfe des im textitSystemkatalogservice gespeicherten Metadaten, die Anfragen durch den Anfrageoptimierungsservice optimiert. Dabei entsteht ein Ausführungsplan, welcher vom Anfrageverarbeitungsservice ausgeführt wird. Dabei kann auf den *Zugriffspfadservice* zurückgegriffen werden, welcher spezielle Zugriffspfadstrukturen verwaltet. Der lesende und schreibende Zugriff auf die verschiedenen Datenobjekte wird durch den Datenobjektservice zur Verfügung gestellt. Dabei werden die Datenobjekte welche in speziellen Datensätzen verwaltet werden, mit Hilfe des *Transaktionsmanagerservice*, in einem *Storageservice* koordiniert abgelegt. Um die spezielle Funktionalität die bei Database-as-a-Service benötigt werden breit zu stellen, werden diese Basisservices um erweiterte Services ergänzt, welche in dem folgenden Kapitel beschrieben werden.

Erweiterte Services

Die erweitertenden Services, welche in Abbildung 5 dargestellt sind, bestehen aus dem *Dienstgüte-, Abrechungs-, Mandantenfähigkeits-, Datenvertraulichkeits-, Integritäts- und Integrationsservice*. Dabei können diese Services auf die Funktionalität der Basisservices zurückgreifen. Der *Dienstgüteservice* dient der Durchsetzung und der Protokollierung des Erfüllungsgrades der Dienstgütegarantien, welche in den Service-Level-Agreements (SLA) festgelegt wurden. Dazu dient der *SLA-Service*, der die eingehenden Anfragen dynamisch mit einer Priorität versieht, welche in den SLA festgelegten Dienstgütegarantien beschrieben wird. Diese Garantien sind im *SLA-Metadatenservice* als Metadaten gespeichert. Vor der Ausführung muss die Operation durch den *Zulassungskontrollservice* freigegeben werden. Bei der Zurückstellung der Operation, wird durch den *Ausführungsplanservice* ein neuer Zeitpunkt, abhängig von der Priorität festgelegt. Die Ausführung der jeweiligen Operationen wird vom *Ausführungskontrollservice* überwacht. Der *Abrechungsservice* verwaltet die Kosten die durch die Ausführung entstehen. Dabei ist das Abrechnungsmodell im *Abrechnungsmetadatenservice* hinterlegt. Der *Nutzungstrackingservice* misst den entstandenen Aufwand und der *Aufwandsbewertungsservice* kann vor der Ausführung eine Komplexitätsbewertung durchführen. Da eine Unterstützung von mandantenfähigen Anwendungen auf Datenbanksystemebene möglich ist, gibt es den *Mandantenfähigkeitsservice*. Dieser speichert zusätzlich zu dem Basisschema in *Systemkatalogservice* eine mandantenspezifische Schemaerweiterung in dem *Erweiterungsmetadatenservice*. Diese wird durch den *Schemamappingservice* in ein gemeinsames Schema umgewandelt. Für Datenschutz und Datensicherheit gibt es den *Datenvertaulichkeits- und Integritätsservice*. Diese werden meisten an die Vor- bzw. Nachverarbeitung angehangen und stellen mit dem *Anfrageübersetzungs- und Anfragenachver-*

arbeitungsservice Dienste zur Verfügung, die eine Übersetzung eingehender Anfragen und Nachverarbeitung von Ergebnissen bereitstellen. Der *Ver- und Entschlüsselungsservice* stellt spezielle Dienste zur Ver- und Entschlüsselung der Daten bereit. Des Weiteren gibt es einen *Signaturservice*, der mit Hilfe des *Signaturverwaltungsservice* Daten signieren kann und die Verwaltung notwendiger Metadaten übernimmt. Als letztes wird der *Integrationsservice* vorgestellt. Da die zu speichernden Daten auf Zugriffs - und Speicherservices verschiedener Anbieter zurückgreifen können und diese nicht zwangsläufig das selbe Datenmodell verwenden, existiert dieser Service. Dazu stellt es den *Schemametadatenservice* zur Verfügung welches die Speicherung eines globalen logischen Schemas übernimmt und Informationen über die Datenlokalität enthält. Es bietet weiterhin einen *Integrationsservice* welches eingehende Anfragen in Anfragen für verschiedene Zugriffsservices übersetzt.[Lan]

Im Anschluss wird eine Adaption des Modells an einen Forschungsprototypen und Amazon Simple DB genauer erläutert.

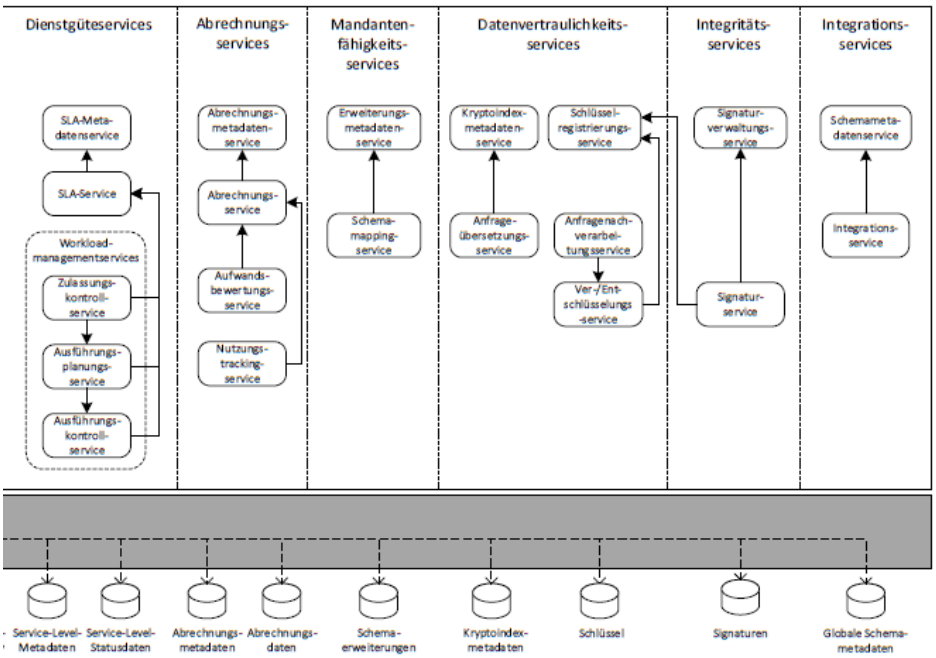

Abbildung 5: erweiterten Services

4.1 Adaption des Database-as-a-Service Modells am einen Forschungsprototypen

Der Forschungsprototype stellt eine Implementierung eines verteilten Datenbanksystems auf Basis von Amazon Simple Storage Service (S3) und Amazon Simple Queue Service (SQS) dar. Dieses

Datenbanksystem soll die Speicherung und den Abruf von Daten ermöglichen, wobei S3 eine Speicherung schlüsselindexierter BLOB's ermöglicht. Der Simple Queue Service dient zur Transaktionsverwaltung durch Message-Queues. Die übrige Funktionalität wird durch eine clientseitge Softwareschicht zur Verfügung gestellt. Dabei besteht ein Datensatz aus einer beliebigen Bytefolge und einem Schlüssel und die Datensätze werden als Kollektion angelegt. Die interne Darstellung wird als B+ Baum realisiert, welche in S3 gespeichert wird. Ebenfalls können sekundäre B+ Bäume angelegt werden, welche Attribute indexieren. Auf die einzelnen Datensätze kann durch Angabe von Schlüssel- oder Attributwerten, unter Angabe der Kollektion, zugegriffen werden. Schreiboperationen werden über SQS koordiniert und jeder Schreiboperation werden zwei SQS-Queues zugeordnet. Die erste dient der Speicherung von Sperrtoken und die zweite dient der Protokollierung der Änderungen. Dabei wird durch den Client zuerst die Sperrtoken und dann die Protokollierung abgerufen. Bei Erfolg der Transaktion wird der Protokolleintrag gelöscht und die Sperrtoken wieder freigegeben. In Abbildung 6 ist der Prototypen dargestellt. Dabei wird der Storageservice durch Amazon Simple Storage Service realisiert. Der im Transaktionsservice befindliche Transaktionsmanagerservice dient der Protokollierung der durchzuführenden Operationen und Verwaltung von Sperren. Dieser wird durch SQS realisiert. Die Datensatzverwaltung und die Verwaltung sekundärer Zugriffsstrukturen wird als clientseitige Funktionalität realisiert. Die Abbildung dieser erfolgt durch den Datenobjekt- und Zugriffspfadservice. Da diese mittels einer gemeinsamen Schnittstelle gekapselt sind, können sie als Service nach außen verwendet werden. Da alle Anfragen an diesen Service gerichtet sind gibt es keine weiteren Daten- und Zugriffsservices.

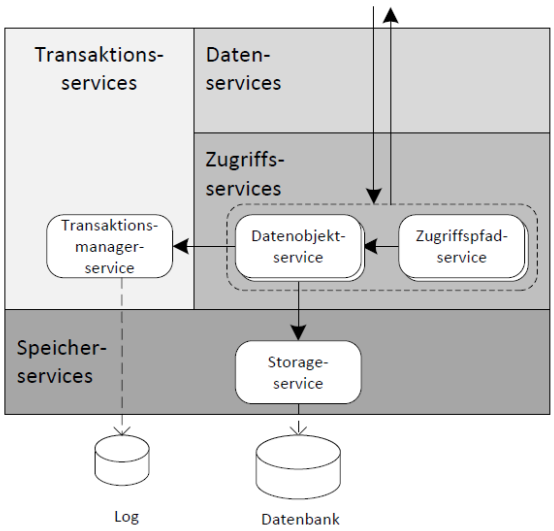

Abbildung 6: Adaption des Database-as-a-Service Modells an einen Forschungsprototypen

4.2 Adaption des Database-as-a-Service Modells an Amazon Simple DB

Amazon Simple DB ist ein Webservice zur Speicherung untypisierter Multi-Maps. Diese Multi-Maps werden als Items bezeichnet. Der Service stellt Einfüge-, Modifizier-, Lösch- und Anfrageoperationen zur Verfügung. Dabei werden die gespeicherten Werte automatisch indexiert. Die Anfragen werden mittels Simple-DB spezifischer Anfragesprache realisiert und der Zugriff wird nur für authentifizierte und autorisierte Nutzer gewährt. Simple-DB verfolgt bei der Sicherstellung der Konsistenz den Ansatz der *Eventual Consistency*. Das bedeutet das ein konsistenter Zustand gesichert ist, nur der Zeitpunkt an dem dies geschieht ist unbekannt. Schreibende Operationen gehen immer genau auf ein Item. Dabei existieren keine Sperrmechanismen, was zur Folge hat, dass die Transaktion mit dem jüngsten Zeitstempel Gültigkeit erlangt. Dabei erfolgt die Abrechnung nutzungsbasiert anhand der Anzahl ausgeführter Operationen und der übertragenen Datenmenge. In der Abbildung 7 ist die Amazon Simple DB dargestellt. Der Schnittstellenservice kapselt die internen Services und ist nach außen sichtbar. Dabei ist er der einzige Service der extern erreichbar ist. Ankommende Transaktionen werden vor der Ausführung durch den Authorisierungsservice authentifiziert und autorisiert. Dabei gehen schreibende Transaktionen an den Aktualisierungsservice. Die eingegangenen Anfragen werden anschließend an den Anfrageoptimierungsservice weitergeleitet, optimiert und in einen Ausführungsplan überführt. Der Anfrageverarbeitungsservice führt dann den Ausführungsplan aus. Der Datenobjektservice ermöglicht den Zugriff auf Items. Die bei Aktualisierungen automatisch erstellten Indexe werden im Zugriffspfadservice verwaltet und können bei der Anfrageverarbeitung mit einbezogen werden. Auf der Ebene des Speicherservices existieren mehrere Instanzen des Storageservices, da Daten repliziert gespeichert werden. Da eine nutzungsbasierte Abrechnung vorliegt, beschränkt sich der Abrechnungsservice auf einen zentralen Abrechnungs-, Abrechnungsmetadaten- und Nutzungstrackingservice.[Lan]

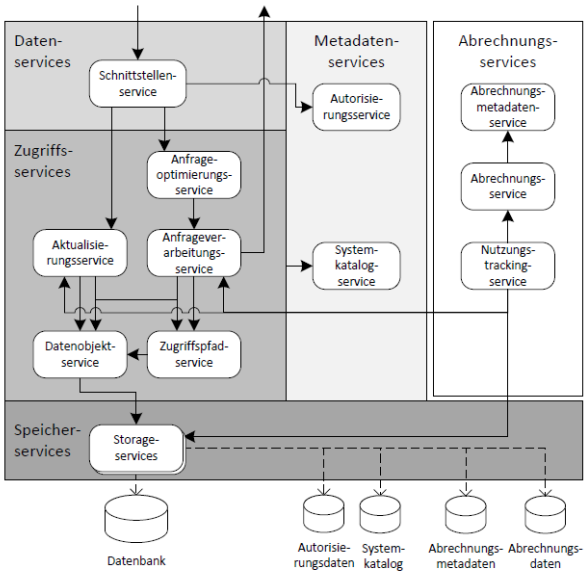

Abbildung 7: Adaption des Database-as-a-Service Modells am Amazon Simple DB

5 Zusammenfassung

Zusammenfassend ist zu sagen, dass Database-as-a-Service einige Vorteile im Vergleich zu lokal installierten Datenbanken bringen. Zum einen werden dadurch die Kosten für die Anschaffung und den Betrieb von Datenbanken stark reduziert. Dadurch kann sich das Unternehmen auf das Kerngeschäft konzentrieren und bezahlt nur die Leistungen die es wirklich benötigt hat. Ein weiterer Vorteil besteht darin, dass bei Database-as-a-Service eine große Skalierbarkeit gegeben ist, das heißt es werden je nach bedarf Ressourcen hinzugefügt oder entzogen, ohne das der Kunde dies mitbekommt. Auch die Administration der Datenbank entfällt für den Kunden, da sich darum die Anbieter kümmern müssen.

Aber trotz aller dieser Vorteile gibt es auch einige Nachteile die gegen einen Einsatz von Database-as-a-Service sprechen. Dabei ist vor allem zu nennen, das die Unternehmensdaten in einem fremden Unternehmen liegen und man dadurch ein gutes Vertrauensverhältnis zu dem Anbieter haben muss. Ebenfalls sind noch nicht alle Datenschutz- und Datensicherheitsaspekte geklärt.

Das Modell welches in Kapitel 5 vorgestellt wurde, bietet eine Lösung an um die verschiedensten Herausforderungen, wie Mandantenfähigkeit und Datenschutz und Datensicherheit zu lösen. Einige dieser im Modell beschriebenen Aspekte wurden aber bei der Implementierung bei dem Prototypen und anhand von Amazons Simple DB nicht umgesetzt. Da aber gerade die Mandantenfähigkeit und Datenschutz und Datensicherheit ein wichtiger Aspekt bei Database-as-a-Service ist sollte man diese Implementierungen nicht für Unternehmenskritische Daten nutzen.

Abschließend ist zu sagen, das sich Database-as-a-Service für einige Anwendungsfälle lohnen würde, wie z.B. Produktkatalog eines Onlineshops. Aber bei kritischen Unternehmensdaten wie Kreditkartendaten ist zu überlegen ob man auf Database-as-a-Service setzen sollte.

6 Literaturverzeichnis

Literatur

[AFG+09] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the clouds: A berkeley view of cloud computing. Technical report, University of California at Berkeley, February 2009.

[ASS+09] Ashraf Aboulnaga, Kenneth Salem, Ahmed A. Soror, Umar Farooq Minhas, Peter Kokosielis, and Sunil Kamath. Deploying database appliances in the cloud. *IEEE Data Eng. Bull.*, 32(1):13–20, 2009.

[BFG+09] Matthias Brantner, Daniela Florescu, David Graf, Donald Kossmann, and Tim Kraska. Building a Database in the Cloud. Technical Report, ETH Zurich, 2009., 2009.

[FH00] et al. Fred Hoch. Software as a service: Strategic backgrounder. 2000.

[KK] Fabian Panse Norbert Ritter Kathleen Krebs, Marc Holze. Konfiguration und Spezifikation bedarfsgerechter Dienstleistungen zur Datenverwaltung. DaaS-Workshop, Münster, 3. März 2009.

[Lan] Jens Lansing. Database-as-a-Service: Charakterisierung und Modellentwurf. DaaS-Workshop, Münster, 3. März 2009.

Abbildungsverzeichnis

Tabellenverzeichnis