Dr. Veit Köppen
(E-Mail: vkoeppen@ovgu.de)
ist geschäftsführender Leiter des Center for Digital Engineering an der Otto-von-Guericke-Universität Magdeburg. Seine Forschungsinteressen liegen im Bereich der Geschäftsprozesse, Business Intelligence sowie des Digital Engineerings.

Francis Gropengießer
(E-Mail: francis.gropengiesser@tu-ilmenau.de)
arbeitet als wissenschaftlicher Mitarbeiter am Fachgebiet Datenbanken und Informationssysteme der Technischen Universität Ilmenau. Seine Forschungsschwerpunkte liegen im Bereich des Transaktionsmanagements für kooperative Systeme und Cloud-Umgebungen.

Dr.-Ing. Martin Kuhlemann
(E-Mail: martin.kuhlemann@ovgu.de)
arbeitet als wissenschaftlicher Mitarbeiter am Lehrstuhl Datenbanken des Instituts für Technische und Betriebliche Informationssysteme der Otto-von-Guericke-Universität Magdeburg. Seine Forschungsinteressen liegen in den Bereichen Software-Produktlinien, Refactoring, Modularisierungstechniken sowie Datenbanken.

Prof. Dr. Gunter Saake
(E-Mail: gunter.saake@ovgu.de)
hat den Lehrstuhl für Datenbanken an der Otto-von-Guericke-Universität Magdeburg inne. Seine Forschungsinteressen umfassen u. a. Design und Abfragesprachen in Datenbanken, Semantik und Spezifikation von objektorientierten Sprachen.

Prof. Dr. Kai-Uwe Sattler
(E-Mail: kus@tu-ilmenau.de)
ist Lehrstuhlinhaber am Fachgebiet Datenbanken und Informationssysteme der Technischen Universität Ilmenau. Zu seinen Forschungsinteressen zählen semantische Interoperabilität, große verteilte Datenbanken und Self-Tuning.

Datenbanken in der Cloud – Transaktionsmanagement für Database as a Service

Die Verlagerung von IT-Lösungen in die Cloud verspricht eine hohe Effizienzsteigerung und Kosteneinsparung. Jedoch muss die Frage beantwortet werden, ob jede benötigte Funktionalität auch in der Cloud realisierbar ist. Erste Cloud-basierte Datenbanksysteme versprechen für viele Anwendungsdomänen hohe Potenziale. Dabei muss zwischen bekannten Eigenschaften und Funktionen wie Konsistenz, Verfügbarkeit und Fehlertoleranz ein Gleichgewicht gefunden werden. In diesem Artikel beleuchten wir typische transaktionale Funktionalitäten im Cloud-Umfeld.

ACID-Eigenschaften in der Cloud

Die Cloud ist in aller Munde. Sie stellt eine konsequente Weiterentwicklung der IT-Servicefunktionalität dar und fokussiert IT und Fachbereich auf die jeweiligen Kernkompetenzen. So führt dies zu einem effizienteren und kostengünstigeren Ansatz in der Nutzung von Ressourcen. Eine Entscheidung für oder wider die Cloud sollte in Abhängigkeit der Anforderungen an die Geschäftsprozesse erfolgen. Getreu dem Motto *IT follows Function* müssen die Domänenanforderungen beachtet werden.

Im Nachfolgenden widmen wir uns der Thematik von der Verwendung von Datenbankdiensten. Aufgrund des adaptiven Ressourcenmanagements bietet die Cloud Vorteile, die für die Nutzung von operativen aber auch analytischen Informationssystemen einen interessanten und vielversprechenden Ansatz bieten.

Im Vordergrund stehen hierbei häufig Kosteneinsparpotenziale. Jedoch gilt es bei der Verwendung von Datenbanken, eine Vielzahl von Restriktionen zu beachten. Wir beschränken uns im Folgenden auf Transaktionen im Datenbankmanagementsystem. Weitere Anforderungen können zum Beispiel aus den Regeln für Online Transactional Processing (OLTP) oder Online Analytical Processing (OLAP) nach Codd [Cod82] resultieren.

Bei Datenbanksystemen erwünschte Eigenschaften werden durch ACID-Anforderungen beschrieben [Här83]. ACID steht hierbei für Atomarität, Konsistenzerhaltung, Isolation und Dauerhaftigkeit.

Die Atomarität von Datenbankoperationen zielt auf eine transaktionale Verarbeitung. Das heißt, entweder wird die Operation ganz oder gar nicht ausgeführt. Integritätsbeziehungen sichern einen konsistenten Datenbestand nach Ausführung einzelner Operationen. Die Anforderung der

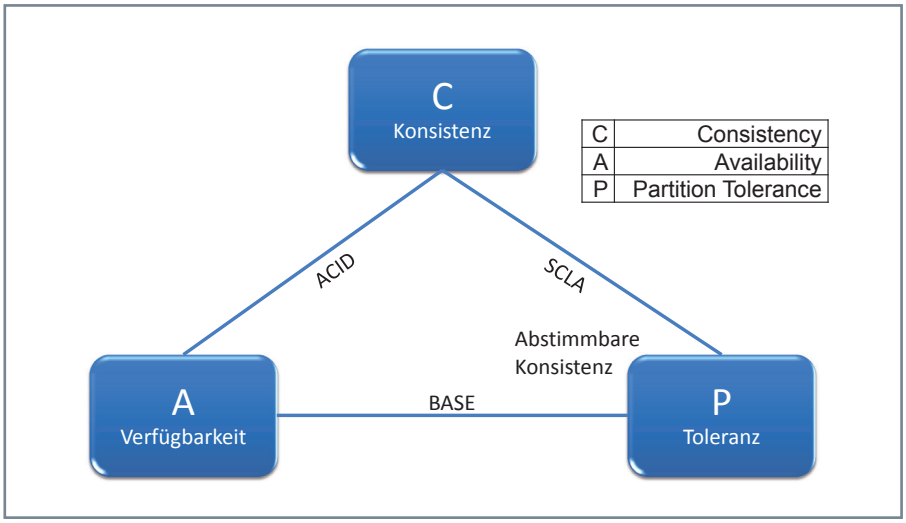

Abb. 1: CAP-Theorem und -Eigenschaften

Isolation schränkt die Beeinflussung nebenläufiger Operationen ein. Mit dem Prinzip der Dauerhaftigkeit wird nicht nur garantiert, dass die Daten dauerhaft in der Datenbank abgelegt sind, sondern auch in Systemfehlerfällen (hardware- und softwareseitig) weiterhin gespeichert vorliegen.

Aufgrund der sich gegenseitig beeinflussenden ACID-Faktoren in verteilten Umgebungen, wie sie durch Servicekapselung und Skalierbarkeit in der Cloud auftreten können, formuliert das CAP-Theorem [Fox97], siehe Abbildung 1, dass nur zwei von drei Eigenschaften, Konsistenz, Verfügbarkeit und Toleranz hinsichtlich Fehlermanagement in verteilten Data-Sharing-Umgebungen erfüllt sein können.

Bei den in der Cloud angebotenen Datenbankdiensten ist die Konsistenzstufe zugunsten der Verfügbarkeit und Skalierbarkeit reduziert. Dies bedeutet zugleich nur eine schwache Ausprägung des Transaktionsmanagements für Cloud-basierte skalierbare Datenbanklösungen und es bleibt dem Anwendungsentwickler die Implementierung der benötigten Funktionalität überlassen.

Neben den ACID-Eigenschaften bei relationalen Datenbankmanagementsystemen haben sich das BASE-Prinzip (Basically Available Soft-state Eventually Consistency) und SCLA (Strongly Consistent Loosely Available) etabliert. Während bei BASE nicht garantiert ist, dass alle Nutzer zu einem Zeitpunkt den gleichen Datenbestand sehen, erhalten sie auf jeden Fall eine, wenn auch veraltete Version der Daten.

Bei SCLA wird eine stärkere Konsistenzstufe im Vergleich zu BASE geschaffen, zudem ist die Verfügbarkeit verglichen mit ACID höher. Zusätzlich existiert hierzu noch der Ansatz der abstimmbaren Konsistenz und für jede Datenbankoperation kann diese festgelegt werden.

Anforderungen unterschiedlicher Anwendungsdomänen

Die Nutzung von Cloud-basierten Diensten erfolgt nach dem Pay-per-use-Modell und umfasst sowohl Infrastruktur, Speicher als auch Software. Damit ist eine Alternative zu eigenbetriebenen Hard- und Softwarelösungen im kosteneffizienten Bereich möglich.

Auch für Datenbanksysteme gibt es bereits eine breite Palette an Lösungsvorschlägen. Diese reichen von Datenbanken in einer Box, z. B. Amazon RDS, bis hin zu hochskalierbaren Lösungen, wie Amazon SimpleDB.

Die Ausnutzung der Elastizität der Cloud-Infrastruktur führt zu einem ressourceneffizienten Betrieb. Jedoch gehen mit der Skalierbarkeit Konsistenzeinbußen einher. Aufgrund der unterschiedlichen Eigenschaften (ACID, BASE, SCLA und abstimmbare Konsistenz) muss in Abhängigkeit der Domäne das entsprechende Anforderungsprofil an das Datenmanagement gefunden werden.

Wir werden im Folgenden vier repräsentative Anwendungsszenarien beschreiben und diese hinsichtlich ihrer ACID-Anforderungen eingruppieren.

Der Online-Warenkorb

In einem Online Store zählt der personalisierte Warenkorb zu den Standardkundenanforderungen. Dieser Warenkorb soll für den Kunden stets verfügbar und auch das Hinzufügen und Löschen von Positionen soll möglich sein. Die ACID-Anforderungen sind gering: aufgrund der einfachen Änderungen im Warenkorb ist die Atomarität hier beschränkt und auch das Konsistenzniveau muss nicht strikt sein, denn es reicht aus, wenn der Kunde den Warenkorb zum jeweiligen Bestellvorgang sieht. Da jeder Nutzer seinen eigenen Warenkorb hat, ist die Isolation ebenfalls nur in sehr geringem Maße von Interesse. Die Dauerhaftigkeit der Daten soll aber über den gesamten Prozess gesichert sein.

ACID-Anforderungen				
Anwendung	**A**	**C**	**I**	**D**
Warenkorb	Pro Operation	Abschließend	Nicht erforderlich	Erforderlich
Indizierung	Verteilt	Strikt	Erforderlich	Erforderlich
Internet Blog	Feingranular	Strikt	Erforderlich	Flexibel
Online Rollenspiele	Konsens	Read-your-Write	Bedingt erforderlich	Erforderlich

Tab. 1: ACID-Anforderungen unterschiedlicher Szenarien

Dienstbasierte Indexstrukturen in Datenbanken

Der effiziente Zugriff auf große Datenbestände wird mittels Indexstrukturen ermöglicht. Auch für Cloud-basierte Datenbanken bieten sich hier verteilt gespeicherte Indexstrukturen an, die nicht notwendigerweise auf denselben Knoten liegen. Eine Änderung im Datenbestand erfordert dann auch eine Änderung in der Indexstruktur.

Somit zieht eine logische Änderung mehrere physische Änderungen nach sich, die atomar ausgeführt werden müssen und deshalb verteilte Transaktionen erfordern. Im Mehrbenutzerbetrieb müssen darüber hinaus Synchronisationsmechanismen implementiert sein, um die notwendige Konsistenz zwischen Datenbestand und Index zu wahren. Auch die Anforderung an die Dauerhaftigkeit der Indexstruktur liegt vor, da bei einem Ausfall die angestrebte Anfrageperformanz nicht gewährleistet werden kann.

Internet-Blog

In kooperativen Anwendungsumgebungen wie einem Internet-Blog existieren vielfältige ACID-Anforderungen. Die Atomarität muss flexibel gestaltbar sein. Ein lang andauernder Arbeitsprozess muss in atomare Teilschritte zerlegbar sein. Dies bedeutet einen geringen Aufwandsverlust im Fehlerfall.

Die oft umfangreichen Operationen der Anwendungsebene müssen durch atomare Ausführung primitiver Datenoperationen realisiert werden. Somit ist ein umfangreiches Transaktionsmanagement erforderlich. Die Konsistenzanforderungen sind eher strikt, da in kooperativen Umgebungen jeder Nutzer wissen muss, was der andere macht. Auch wird häufig auf denselben Daten gearbeitet und daher ist ein

System	Atomarität	Konsistenz	Isolation	Dauerhaftigkeit	Prinzip
Amazon RDS	Transaktional	Strikt	ANSI SQL	Strikt	ACID
Amazon S3	Pro Operation	Abschließend	Last-Write-Wins	Replikate	BASE
Amazon SimpleDB	Mehrere Operationen	Abschließend, Strikt	Optimistisch	Replikate	BASE
Azure BLOB	Mehrere Operationen	Strikt	Schreib-sperren	3 Replikate	BASE
Azure SQL	Transaktional	Strikt	ANSI SQL	Strikt	ACID
Azure Table	Transaktionen	Strikt pro Partition	Optimistisch	3 Replikate	BASE
Cassandra	Mehrere Operationen	Konsistenzstufen	Zeitstempel	Einstellbar	Anpassung
Riak	Pro Operation	Abschließend	Vector Clocks	Einstellbar	BASE

Tab. 2: ACID-Eigenschaften ausgewählter Systeme

Einsatz semantischer Synchronisationsmodelle notwendig. Bei der Dauerhaftigkeit sind unterschiedliche Anforderungsprofile in Abhängigkeit der Prozesse möglich.

Online-Rollenspiele

Im Bereich der Online-Rollenspiele entwickeln und verändern sich Informationen über Spielwelt und Charaktere dynamisch und müssen allen Teilnehmern zur Verfügung stehen. Jedoch müssen nicht alle Informationen mit strikter Konsistenz verteilt werden. Zum Beispiel kann es unwichtig sein, ob ein Stein von Position A nach Position B bewegt wurde. Es ist ausreichend, Änderungen durch ein Konsensverfahren zu verteilen.

Essentielle Spielaktionen, zum Beispiel in Wettkampfbedingungen, müssen andererseits sofort und konsistent propagiert werden. Jeder Spieler sollte zumindest seine eigenen Aktionsergebnisse unmittelbar sehen (Read-your-Writes). Häufig erfolgt eine interaktive Synchronisation bei konkurrierenden Änderungen. Diese sollten jedoch dauerhaft gespeichert sein.

Die unterschiedlichen Szenarioanforderungen an die ACID-Eigenschaften werden in Tabelle 1 zusammengefasst.

ACID-Garantien in bestehenden Platform as a Service-Lösungen

In diesem Abschnitt wollen wir die folgenden ausgewählten Dienste hinsichtlich der ACID- Eigenschaften kurz darstellen und diskutieren:

- Amazon RDS, S3, und SimpleDB,
- Azure BLOB, SQL und Table,
- Cassandra und Riak.

Eine detaillierte Darstellung der Systeme hinsichtlich der ACID-Eigenschaften ist in Tabelle 2 abgebildet.

Die nach dem ACID-Prinzip arbeitenden „Database-in-a-Box“-Lösungen können über JDBC/ODBC angesprochen werden. Alle BASE-Systeme lassen sich über die REST-Schnittstelle einbinden. REST (Representational State Transfer) stellt dabei ein geläufiges Programmierparadigma in der Domäne der Webanwendungen dar. Cassandra bietet nur eine Apache-Thrift-Schnittstelle.

Hinsichtlich der Atomarität der Datenoperationen sind alle Systeme bezüglich Anzahl oder Ausdehnung beschränkt. Eine Erweiterung kann nur über externe Transaktionsmodelle erfolgen. Eine abschließende Konsistenz bzw. eine abgeleitete Form davon muss ebenfalls auf Anwendungsebene sichergestellt werden, um eine strikte Konsistenz zu ermöglichen.

Die Isolation wird nur durch grundlegende Protokolle in Teilen erreicht, sodass auch hier eine Erweiterung durch externe Synchronisationsprotokolle zur Erreichung strikter Isolation notwendig ist. In Cloud-basierten Lösungen werden Replikationsmechanismen eingesetzt. Dies beinhaltet aber nur einen Teil für die Sicherstellung von Dauerhaftigkeit. Es müssen zusätzlich Recovery-Protokolle für den Fehlerfall extern bereitgestellt werden. Eine mögliche Architektur für Cloud-basierte Datenbanklösungen ist in Abbildung 2 dargestellt.

Bei der Auswahl eines Dienstes müssen die durch die fachliche Domäne vorgegebe-

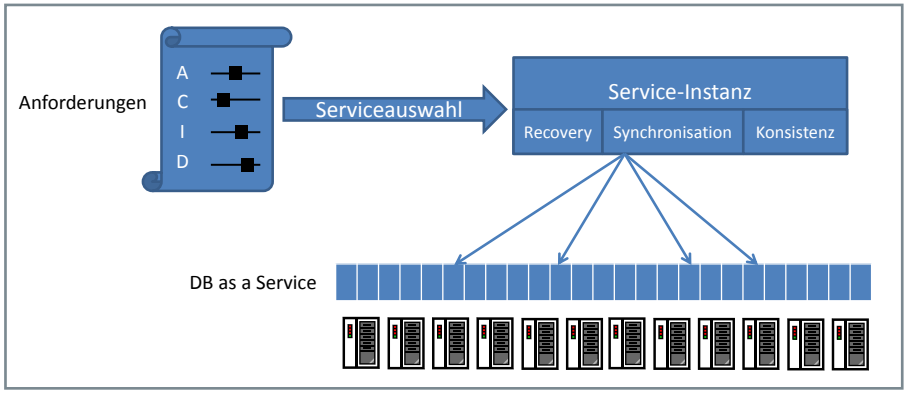

Abb. 2: Cloud-basierte Datenbanksystemarchitektur

nen Anforderungen beachtet werden. Die Auswahl und Adaption der Dienste bzw. die Orchestrierung muss dabei konkrete Verfahren von Commit-, Synchronisations- und Recovery-Protokollen umfassen.

Die Cloud-spezifischen Besonderheiten hinsichtlich Infrastruktur, Laufzeitumgebung und Speicherdienst müssen insbesondere berücksichtigt werden. Hierzu zählen zum Beispiel:

- Cloud-Monitoring-Dienste, z. B. Amazon CloudWatch, Microsoft Azure Watch und Google AppEngine Dashboard,
- Batch-Operationen für leichtgewichtige Transaktionsmodelle, wie z. B. Amazon SimpleDB und Google AppEngine,
- Bedingte Operationen für optimistische Synchronisationsprotokolle, wie z. B. bei Amazon SimpleDB,
- Nachrichtenwarteschlangen, wie Amazon SQS und Microsoft Azure Queue Service und
- Versionierung, wie z. B. bei Amazon S3 und Google BigTable.

Zusammenfassung

Mit dem Einzug von Datenbanken in die Cloud ergeben sich für Unternehmen und Privatpersonen einfache Möglichkeiten, Daten preiswert zu verarbeiten und zu speichern. Dies kann an die entsprechenden Bedürfnisse angepasst erfolgen. Bevor es jedoch zu einer Auswahl eines Datenbankdienstes kommt, sollten die Anforderungen an die Datenverarbeitung evaluiert werden.

Die Analyse unterschiedlicher Anwendungsszenarien und Dienste hat ergeben, dass ein Transaktionsdienst in der Cloud sinnvoll ist. Dabei müssen jedoch unterschiedliche Herausforderungen wie die Definition von ACID-Stufen, eine dynamische Service-Konfiguration und entsprechende Preismodelle berücksichtigt werden.

Bestehende abgeschlossene Systeme bieten hinsichtlich des Transaktionsmanagements eine Fülle an Funktionalität, die aktuell nur partiell in den Cloud-basierten Lösungen angeboten werden. So kann es notwendig sein, in den entsprechenden Anwendungen zusätzliche Funktionalität zu implementieren, um Anforderungen an Konsistenz und Fehlertoleranz zu begegnen. Momentan existieren noch keine Cloud-Dienste, die funktionale Eigenschaften klassischer Datenbankmanagementsysteme vollständig abbilden können. ■

Literatur & Links

[Cod82] E. F. Codd, Relational database: a practical foundation for productivity, CACM, 25 (2), 109-117, 1982.

[Fox97] A. Fox, S. D. Gribble, Y. Chawathe, E. A. Brewer, P. Gauthier, Cluster-based Scalable Network Services, SOSP, S. 78-91, 1997.

[Här83] T. Härder, A. Reuter, Principles of Transaction-Oriented Database Recovery, Computing Surveys, 15(4), S. 287-317, 1983.

[Moh11] S. Mohammad, Survey and Classification of Data Management in the Cloud, Magdeburg, 2011.

[ama] http://aws.amazon.com/de/s3/

[win] www.windowsazure.com