Grundlagen und Anwendungsbeispiele für das NOSQL-Datenbankmanagementsystem DynamoDB

Seminararbeit

vorgelegt am 25. Januar 2017

Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik

Kurs WWI2014I

von

|  |  |
| --- | --- |
| Betreuer in der Ausbildungsstätte: | DHBW Stuttgart: |
| Prof. Dr. Nikolai Preiß DHBW Stuttgart | [Titel, Vorname und Nachname des wissenschaftl. Betreuers/Prüfers] |

Kerstin Farke  
Dhruv Mahandru  
Henning Mohr  
Käthe Vrettos

**Inhaltsverzeichnis**

[Abkürzungsverzeichnis (bei Bedarf) IV](#_Toc469343345)

[Abbildungsverzeichnis (bei Bedarf) V](#_Toc469343346)

[Tabellenverzeichnis (bei Bedarf) VI](#_Toc469343347)

[1 Einleitung 1](#_Toc469343348)

[2 Theoretische Grundlagen 2](#_Toc469343349)

[2.1 Definitionen 2](#_Toc469343350)

[2.2 Datenbank Prinzipien 2](#_Toc469343351)

[2.2.1 ACID 2](#_Toc469343352)

[2.2.2 BASE 3](#_Toc469343353)

[2.2.3 CAP-Theorem 3](#_Toc469343354)

[2.3 Entwicklungen in der Datenbankbranche 4](#_Toc469343355)

[2.4 Datenbankmodelle 5](#_Toc469343356)

[2.4.1 Relationale Datenbanken 5](#_Toc469343357)

[2.4.2 NoSQL Datenbanken 6](#_Toc469343358)

[2.4.3 Zusammenfassung 8](#_Toc469343359)

[3 DynamoDB 10](#_Toc469343360)

[3.1 Generelle Informationen über AWS 10](#_Toc469343361)

[3.1.1 Infrastructre as a Serivice (IaaS)/Kunden 10](#_Toc469343362)

[3.1.2 Gründe für die Verwendung von AWS/Entwicklungsgeschichte (evtl.) 10](#_Toc469343363)

[3.1.3 Ziele 10](#_Toc469343364)

[3.1.4 Vor- und Nachteile von AWS 10](#_Toc469343365)

[3.2 Was ist DynamoDB 10](#_Toc469343366)

[3.2.1 Database as a Service (DBaaS) Definition 10](#_Toc469343367)

[3.2.2 Geschichte der DynamoDB 10](#_Toc469343368)

[3.2.3 Verwendung der DynamoDB in der Amazon Infrastruktur 10](#_Toc469343369)

[3.2.4 Preismodell 10](#_Toc469343370)

[3.3 Art der Datenverwaltung/Datenmodell 10](#_Toc469343371)

[3.3.1 Tabelle, Elemente und Attribute Konzept 10](#_Toc469343372)

[3.3.2 Eigenschaften (Fully managed, Durable, Scalable, Fast, Simple Administration, Flexible, Fault Tolerance, Indexing, Secure, Cost Effective) 10](#_Toc469343373)

[3.3.3 Datenspeicherung über die SSD 10](#_Toc469343374)

[3.3.4 Key-Value Eigenschaft 10](#_Toc469343375)

[3.3.5 Datentypen 10](#_Toc469343376)

[3.4 Vergleich mit anderes NoSQL-Lösungen 10](#_Toc469343377)

[3.4.1 Microsoft Azure 10](#_Toc469343378)

[3.4.2 Redis 10](#_Toc469343379)

[3.4.3 Vorteile und Nachteile von DynamoDB 10](#_Toc469343380)

[4 Praxis 11](#_Toc469343381)

[4.1 Insgesamt zum Zurechtfinden 11](#_Toc469343382)

[4.2 Testdatenbank mit Testdaten 11](#_Toc469343383)

[4.3 Datenbanksprache 11](#_Toc469343384)

[4.4 GUI/API 11](#_Toc469343385)

[4.5 Ablageform der Daten 11](#_Toc469343386)

[5 Schluss 12](#_Toc469343387)

[5.1 Fazit 12](#_Toc469343388)

[5.2 Ausblick 12](#_Toc469343389)

[Anhang (bei Bedarf) 13](#_Toc469343390)

[Quellenverzeichnis 14](#_Toc469343391)

[Ehrenwörtliche Erklärung 15](#_Toc469343392)

# Abkürzungsverzeichnis (bei Bedarf)

AktG = Aktiengesetz

AWS = Amazon Web Services

BFH = Bundesfinanzhof

# Abbildungsverzeichnis (bei Bedarf)

[Abb. 1: Eine Abbildung 2](#_Toc444006772)

# Tabellenverzeichnis (bei Bedarf)

[Tabelle 1: Eine Beispieltabelle 2](#_Toc444006773)

1. Einleitung
2. Theoretische Grundlagen
   1. Definitionen

**GIT TEST**

* Datenbank
* Datenbankmanagementsystem
* Database as a Service
  1. Datenbank Prinzipien
     1. ACID
* Atomarität 🡪 jede Transaktion erfolgt atomar, also zusammenhängend
* Konsistenz
* Isolation
* Dauerhaftigkeit
* 🡪 geben wieder, was wir von Transaktionsbetrieb erwarten
* Entweder die gesamte Transaktion wird ausgeführt, oder die Transaktion findet gar nicht statt, auch nicht in Teilen
* Zwei wichtige Begriffe:
  + COMMIT;
  + ROLLBACK;

(Schicker, Edwin – Datenbanken und SQL: Eine praxisorientierte Einführung mit Anwendungen in Oracle, SQL – Seite 18)

* Atomarität: „everything or nothing“ – Wird ein Teil der Transaktion nicht komplett durchgeführt, schlägt die ganze Transaktion fehl
* Konsistenz: Stellt sicher, dass die Datenbank vor und nach jeder Transaktion in einem soliden Status ist
* Isolation: stellt sicher, dass mehrere Transaktionen, welche zur selben Zeit ausgeführt werden, nicht die jeweils anderen Transaktionen beeinflussen – dadurch müssen gleichzeitig ausführbare Transaktionen in „serialized“ sein
* Dauerhaftigkeit: Stellt sicher, dass sobald eine Transaktion durchgeführt und „commited“ worden ist, sie permanent den gleichen Status behält – z.B. permanent gespeichert wird, auch wenn es Fehler gibt oder das System crashed

(Mahamed, Mohamed A; Altrafi, Obal G; Ismail, Mohammed O – Relational vs. NoSQL Databases: A Survey – Seite 598)

* Kommt von der SQL Umgebung
* In NoSQL benutzen wir nicht das ACID Konzept. Grund: Konsistenz-Feature von SQL

(Sharma, Vatika; Dave, Meenu – SQL and NoSQL Databases – Seite 21)

* + 1. BASE
* B = Basically
* A = Available
* S = Soft State
* E = Eventual consistency
* Gegenteil von ACID
* NoSQL Datenbanken bestehen zwischen ACID und BASE
* Nach einer Transaktions-Konsistenz, besteht ein „soft“ und nicht „solid“ state
* Hauptfokus: permanente Verfügbarkeit

(Sharma, Vatika; Dave, Meenu – SQL and NoSQL Databases – Seite 21)

* Konsistenz nach einer Abfrage ist nicht mehr in einem “solid state”, sondern in einem “soft state”
* Konsistenz soll nicht direkt nach Ende einer Transaktion erreicht werden, sondern nach Ende einer Operation
* Fokus ist permanente Verfügbarkeit
* Gegensatz von ACID
* NoSQL Datenbanken existieren in dem Spektrum zwischen ACID und BASE

(Mahamed, Mohamed A; Altrafi, Obal G; Ismail, Mohammed O – Relational vs. NoSQL Databases: A Survey – Seiten 598 - 599)

* + 1. CAP-Theorem
* Erste Erscheinung im Jahr 2000, vorgestellt von Eric Brewer
* Consistency: Die Daten sind auf jeder Repliation auf jedem Server immer gleich
* Availability: Daten müssen immer zugänglich sein (permanente Verfügbarkeit=
* Partition Tolerance: Die Datenbank funktioniert trotz Network oder Maschinen failure
* Das Theorem sagt aus, dass nur 2 dieser 3 aspekte in einem verteilten System ermöglicht werden können – man muss sich also 2 „aussuchen“

(Mahamed, Mohamed A; Altrafi, Obal G; Ismail, Mohammed O – Relational vs. NoSQL Databases: A Survey – Seite 599)

* CAP folgt 3 Prinzipien
  + 1. Die verfügbaren Daten sollten auf allen Maschinen gleich sein und Updates sollten frequent durchgeführt werden
  + 2. Daten müssen permanent verfügbar sein und sollten jedes mal zugänglich sein
  + 3. Währen eines „machine failures“ oder anderen Fehlern der Maschine, sollte die Datenbank gut weiterlaufen, ohne die Arbeit zu stoppen

(Sharma, Vatika; Dave, Meenu – SQL and NoSQL Databases – Seiten 21 – 22)

* 1. Entwicklungen in der Datenbankbranche
* Relationale Datenbanken wurden in den 70er Jahren entwickelt, um strukturierte Daten in einer organisierten Art und Weise in Tabellen zu speichern.
* Hat eigene Abfrage Sprache – SQL (Structured Query Language)
* Daten wachsen immer mehr 🡪 relationale Datenbanken haben dadurch einige Einschränkungen
  + Datenabfragen sind nicht mehr so effizient durch das große Volumen von Daten
  + Speicherung und Management von größeren Datenbank wird schwierig

(Abramova, Veronika; Bernadino, Jorge; Furtado, Pedro – Experimental Evaluation of NoSQL Databases – Seite 1)

* E.F. Codd erfand relationale Datenbanken in 1970

(Jatan, Nishta; Puri, Sahil; Ahuja, Mehak; Kathuria, Ishita; Gosain, Dishant – A Survey and Comparison of Relational and Non-Relational Database – Seite 2)

* Relationale Datenbanken, basierenden auf dem relationalen Modell, gibt es bereits seit mehr als 30 Jahren – zu der Zeit hauptsächlich mit der Verantwortlichkeit der Datenverarbeitung
* Heute hauptsächlich: Speicherung von Informationen von finanziellen Informationen, persönlichen Daten und mehr
* Viele Unternehmen haben begonnen nicht-relationale Datenbanken in ihren Betrieben einzusetzen – auch bekannt unter dem Namen NoSQL Datenbanken
* NoSQL wurde als erstes in den 1998er von Carlo Strozzi genannt – dieser entwickelte eine file-based Datenbank
* Erst um 2009 wurde es ein wahrer Konkurrent zu relationalen Datenbanken

(Mahamde, Mohamed A; Altrafi, Obay G; Ismail, Mohammed O. – Relational vs. NoSQL Databases: A Survey – Seite 598)

* Nicht-relationale Datenbanken gab es vor der Entdeckung des Relationenmodells durch Ted Codd in Form von hierarchischen oder netzwerkartigen Datenbanken
* Nach Aufkommen von relationalen Datenbanksystemen wurden nicht-relationale Ansätze weiterhin für technische oder wissenschaftliche Anwendungen genutzt
* Mit dem Aufkommen des Internets oder Vielzahl von webbasierten Anwendungen haben nicht-relationale Datenkonzepte gegenüber relationalen an Gewicht gewonnen

(Meier, Andreas; Kaufmann, Michael – SQL- & NoSQL-Datenbanken – Seite 18)

* Verwaltung riesiger Datenbestände im Internet wird immer wichtiger und auch marktentscheidend
* Normale relationale Datenbanken können diese Fülle von Anfragen nicht mehr effizient bewältigen
* Relationale Datenbanken sind auch extreme Sicherheit bedacht, und der Transaktionsmechanismus ist eine wesentliche Säule dieser Datenbank

(Schicker, Edwin – Datenbanken und SQL: Eine praxisorientierte Einführung mit Anwendungen in Oracle, SQL – Seiten 14 – 15)

* 1. Datenbankmodelle
     1. Relationale Datenbanken
* Definition
  + Werden durch ACID Prinzipien repräsentiert
  + Wenn Konsistenz von hoher Relevanz ist, sollten relationale Datenbanken gewählt werden

Abramova, Veronika, Bernadino, Jorge; Furtado, Pedro – Experimental Evaluation of NoSQL Databases – Seiten 1-2

* Vorteile
  + Industriestandard und kann dadurch viel Erfahrung abzeichnen. Außerdem gibt es eine Dokumentation von Professionals, welche in ihrer ganzen Karriere die gleichen Paradigmen genutzt haben
  + Reife von relationalen Datenbanken – es können Jahre von Verbesserungen abgezeichnet werden, welche teilweise von bedeutenden Players wie Oracle, IBM und Microsoft, sowie der Open-Source Community durchgeführt worden sind
  + Hoher Level von Support
  + Gute Wahl wenn die Konsistenz wichtig ist
  + Möglichkeit komplexe Joins zu kreieren, damit Datenbank Administratoren tiefe und komplexe Abfragen implementieren kann, welche bei Dokument und Key-Value Datenbanken nicht umsetzbar wären

Hammes, Dayne; Medero, Hiram; Mitchell, Harrison – Comparison of NoSQL and SQL Databases in the Cloud – Seite 1

* Nachteile
  + Daten müssen definiert werden, bevor man sie speichern kann

Abramova, Veronika, Bernadino, Jorge; Furtado, Pedro – Experimental Evaluation of NoSQL Databases – Seiten 1-2

* + 1. NoSQL Datenbanken
* Definition
  + Basieren auf BASE
  + Alle NoSQL Datenbanken werden durch die fehlenden Relationen zwischen den verschiedenen Records charakterisiert

Abramova, Veronika, Bernadino, Jorge; Furtado, Pedro – Experimental Evaluation of NoSQL Databases – Seiten 1-2

* + Haben an Popularität gewonnen, wegen der einfachen Skalierbarkeit und einer verbesserten Sicherheit
  + NoSQL Implementationen basieren auf Entitäten und unterstützen die Funktionen von RDBMSen wie das Sortieren, Indexen, Projektieren und Abfragen von Daten
  + Transaktionen, die atomare Konsistenz garantieren, werden nicht unterstützt und das updaten der Database über mehrere Entitäten ist ein späterer Prozess
  + Joins und ACID Garantieren werden eingetauscht für eine höhere Transaktionsgeschwindigkeit

Hammes, Dayne; Medero, Hiram; Mitchell, Harrison – Comparison of NoSQL and SQL Databases in the Cloud – Seite 1

* Arten
* Key-Value Store
  + In dieser Datenbank werden alle gelagerten Daten durch ein „key-value pair“ je Data repräsentiert. Das heißt jeder key ist einzigartig und erlaubt es Record Informationen als Wert anzufordern. Diese Struktur ist auch unter dem Namen „Hash Table“ bekannt, wo Datenabrufe normalerweise durch den Key durchgeführt werden

Abramova, Veronika, Bernadino, Jorge; Furtado, Pedro – Experimental Evaluation of NoSQL Databases – Seiten 1-2

* Document Store
  + Datenbanken sind dafür designt die gespeicherten Daten, welche in Dokumenten verschiedener Formate (XML, JSON etc.) zu managen.
  + Komplexer als Key-Value Store

Abramova, Veronika, Bernadino, Jorge; Furtado, Pedro – Experimental Evaluation of NoSQL Databases – Seiten 1-2

* Column Store
  + Struktur ist ähnlich zu den relationalen Datenbanken
  + Alle Daten sind in Reihen und Spalten gespeichert
  + Spalten, welche ähnliche Daten speichern und oft zusammen abgerufen werden, können gruppiert werden

Abramova, Veronika, Bernadino, Jorge; Furtado, Pedro – Experimental Evaluation of NoSQL Databases – Seiten 1-2

* Graphdatenbank
  + Wird genutzt wenn gespeicherte Daten als Graph mit verflochtenen Elementen (wie Social Networking, Raod Maps, Transport Routes) repräsentiert werden und

Abramova, Veronika, Bernadino, Jorge; Furtado, Pedro – Experimental Evaluation of NoSQL Databases – Seiten 1-2

* Vorteile
  + NoSQL bietet ein einfaches Applikation Management und nimmt die Notwendigkeit von Application Änderungen oder Datenbank Schema Änderungen
  + NoSQL Datenbanken bieten eine leichtere, horizontale Skalierbarkeit
  + NoSQL Datenbanken können transparent Nutzen von neuen Clustern und Nodes ziehen, ohne dass das Database Administrations Management oder eine manuelle Distribution von Informationen benötigt wird
  + Horizontale Skalierbarkeit erlaubt es hunderten Tiefpreis Servern Anfragen zu zufrieden zu stellen, während sich die Kosten des Unternehmens verringern
  + NoSQL Datenbanken werden projiziert um automatisch Daten zu managen und zu verteilen, Daten zu von Fehlern zu recovern und ein ganzes System automatisch wiederherzustellen

Abramova, Veronika, Bernadino, Jorge; Furtado, Pedro – Experimental Evaluation of NoSQL Databases – Seiten 1-2

* Nachteile
  + 1. Zusammenfassung

|  |  |  |
| --- | --- | --- |
|  | **Relationale Datenbanken** | **NoSQL Datenbanken** |
| **Transaktionsausfallsicherheit** | Garantiert sehr hohe Transaktionsausfallsicherheit | Spanne von BASE und ACID |
| **Datenmodell** | Daten als Tupel repräsentiert und durch Relationen gruppiert  Die Relationen beinhaltet Tupel (Reihen), mit Attributen (Spalten) Attribut kann durch Domain (Menge von Werten) angesprochen werden | Nehmen Modelliertechniken wie Key-Value Stores, Graphen- oder Dokumenten Stores.  NoSQL Datenbanken kann aus einem oder mehreren Datenmodellen bestehen |
| **Skalierbarkeit** | vertikal | horizontal |
| **Cloud** | Nicht gut für den Cloud-Betrieb geeignet, da sie nicht wichtige Funktionen nicht unterstützen und die nur bis zu einem bestimmten Limit skaliert werden kann | Beste Lösung für Cloud Datenbanken, da alle Charakteristiken, welche NoSQL definieren, in der Cloud Umgebung gewünscht werden |
| **Big Data Handling** | Lösung ist Skalierbarkeit und Datenverteilung in Form von horizontaler oder vertikaler Skalierbarkeit – nur begrenzt möglich | Entworfen um Big Data zu handhaben – vorhandene Methoden erleichtern und verbessern die Performance vom Speichern und Auslesen der Daten |
| **Data Warehouse** | Über Zeit nimmt die Anzahl der gespeicherten Daten stark zu, wodurch ein Big Data Problem entsteht – führt zu Performance Abbau | Design fokussiert sich auf hohe Performance, Skalierbarkeit, Verfügbarkeit und das Speichern von Big Data – dadurch eine gute Lösung für Data Warehouses |
| **Komplexität** | Erhöht sich, da Nutzer Daten in konvertieren müssen, damit sie in die Tabellen passen | Haben die Fähigkeit unstrukturierte und semi-strukturierte Daten zu speichern |
| **Crash Recovery** | Crash Recovery durch den Recovery Manager, welcher dafür verantwortlich ist sicherzustellen, dass Transaktionen atomar und dauerhaft sind [durch log-files und ARIES algorithmus] | Benötigt Replikationen als Backup, um von einem Crash zu recovern |
| **Sicherheit** | Hat sehr sichere Mechanismen um Security Services bereitzustellen – trotzdem gibt es Sicherheitsrisiken wie SQL Injections, Cross Site Scripting, Root Kits etc. | Wurde eher für die Verbesserung von Performance und Big Data auf den Markt gebracht, nicht um Sicherheitslücken zu füllen |

(Mahamed, Mohamed A; Altrafi, Obay G; Ismail, Mohammed O – Relational vs. NoSQL Databases: A Survey – Seite 600)

1. DynamoDB

Das zweite theoretisch ausgelegte Kapitel beschäftigt sich mit der Einführung in die *Amazon Web Services* und stellt im weiteren Verlauf die DynamoDB im Detail vor. Der Einstieg widmet sich den Hintergründen der *Amazon Web Services*, um ein grobes Verständnis für die Infrastruktur zu erzeugen. Anschließend sollen grundlegende Informationen zu DynamoDB hinterlegt werden. In Folge dessen gilt es, die Art der Datenverwaltung und das Datenmodell der NoSQL-Datenbank zu erläutern und zu untersuchen. Abschließend soll ein Vergleich mit alternativen NoSQL-Lösungen die Einschränkungen bzw. die besonderen Merkmale von DynamoDB ausarbeiten.

* 1. Generelle Informationen über AWS

Die DynamoDB ist ein Teil der *Amazon Web Services* und daher ist es relevant, die Merkmale und Besonderheiten der Infrastruktur zu erläutern, bevor eine Einführung in Datenbanklösung gleistet werden kann. Das folgende Unterkapitel ist auf die Einführung in AWS ausgerichtet. Im Verlauf dieses Unterkapitels werden die Ziele und Absichten der Dienstleistung aufgezeigt. Im Anschluss folgt eine Gegenüberstellung der Vorteile und Nachteile der Infrastruktur.

* + 1. Infrastructre as a Serivice (IaaS)/Kunden

Das Modell von *Infrastructure as a Service* stellt laut einem Artikel von Wolfgang Sommergut „die IT-Ressourcen, wie Rechenleistung, [Speicherplatz] oder Netzwerkkapazitäten zur Verfügung“[[1]](#footnote-2). Außerdem habe der Anwender die Chance, das Betriebssystem, sowie die Applikationen eigenständig zu steuern und die Infrastruktur nach eigenem Bedarf zusammenzustellen. Im Wesentlichen handelt es sich bei AWS also um ein *IaaS,* die die Möglichkeiten zur flexiblen Datenspeicherung und den Ausbau der verfügbaren Rechenleistung bereitstellt. Neben den genannten Ausprägungen, bietet AWS eine Vielfalt von Ressourcen an, die in Tabell 1 zusammengefast sind.

|  |  |
| --- | --- |
| **IT-Ressource** | **Beschreibung** |
| Online-Speicher | Es handelt sich hierbei um Cloud-Speicher. |
| Near-line Speicher |  |
| Server |  |
| Nachrichtendienste |  |
| Überwachung |  |
| Datenbanksystem |  |
|  |  |
|  |  |

Tabelle 1 IT-Ressourcen in AWS[[2]](#footnote-3)

* + 1. Gründe für die Verwendung von AWS/Entwicklungsgeschichte (evtl.)
    2. Ziele
    3. Vor- und Nachteile von AWS
  1. Was ist DynamoDB
     1. Database as a Service (DBaaS) Definition
     2. Geschichte der DynamoDB
     3. Verwendung der DynamoDB in der Amazon Infrastruktur
     4. Preismodell
  2. Art der Datenverwaltung/Datenmodell
     1. Tabelle, Elemente und Attribute Konzept
     2. Eigenschaften (Fully managed, Durable, Scalable, Fast, Simple Administration, Flexible, Fault Tolerance, Indexing, Secure, Cost Effective)
     3. Datenspeicherung über die SSD
     4. Key-Value Eigenschaft
     5. Datentypen
  3. Vergleich mit anderes NoSQL-Lösungen
     1. Microsoft Azure
     2. Redis
     3. Vorteile und Nachteile von DynamoDB

1. Praxis

Im Praxisteil werden folgende Themen behandelt!!!!!!!

* 1. Insgesamt zum Zurechtfinden
  2. Testdatenbank mit Testdaten
  3. Datenbanksprache
  4. GUI/API
  5. Ablageform der Daten

1. Schluss
   1. Fazit
   2. Ausblick

Anhang (bei Bedarf)

# Quellenverzeichnis

**Literaturverzeichnis**

**Preiß, N. (2007):** Entwurf und Verarbeitung relationaler Datenbanken, München/Wien: Oldenbourg

**Steger, J. (2006):** Kosten- und Leistungsrechnung, 4. Aufl., München/Wien: Oldenbourg

**Stoi, R. (2003):** Management und Controlling von Intangibles, in: Studium & Praxis, 4. Jg., Nr. 1, S. 34-46

**Verzeichnis der Internet- und Intranetquellen (bei Bedarf)**

**Gesprächsverzeichnis (bei Bedarf)**

# Ehrenwörtliche Erklärung

Wir versichern hiermit, dass wir unsere Seminararbeit mit dem Thema: [ Thema einfügen ] selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben. Wir versichern zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ort, Datum) (Kerstin Farke)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ort, Datum) (Dhruv Mahandru)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ort, Datum) (Henning Mohr)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ort, Datum) (Käthe Vrettos)

1. Artikel aus Computerwochen von Wolfagang Sommergut

   http://www.computerwoche.de/a/was-sie-ueber-die-cloud-wissen-muessen,2504589,2 [↑](#footnote-ref-2)
2. # Entenommen aus Amazon Web Services: Migrating Your .NET Enterprise Application

   von Rob Linton, Seite 8 [↑](#footnote-ref-3)