Amazon SimpleDB
Cloud-Computing Seminar (CLCP)

Nils Weiher

Universität Heidelberg
n.weiher@stud.uni-heidelberg.de

25.5.2009

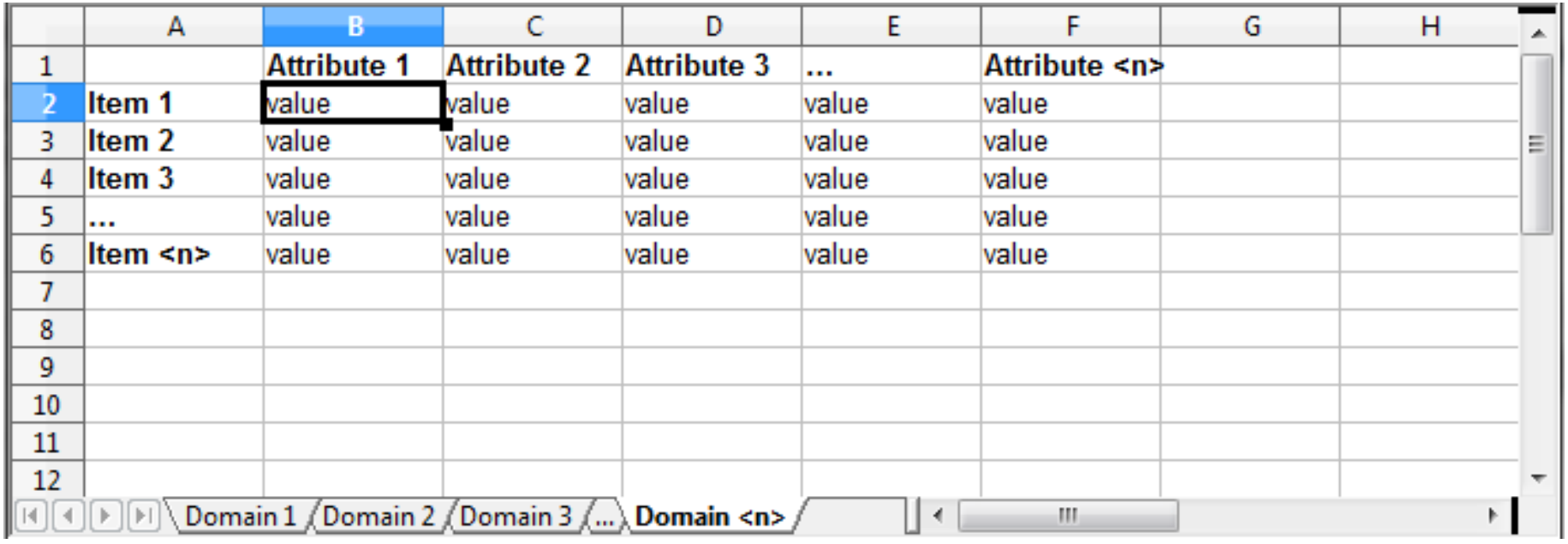
Gliederung

1. Überblick
2. Vorteile, Nachteile und Grenzen
3. Implementierung
4. API
5. Zusammenfassung
6. Quellen

Was ist Amazon SimpleDB? (1)

- ein Web Service
- Teil von AWS (Amazon Web Services)
- seit 1.12.2008 als öffentliche **Beta**
- große Mengen an Datensätzen strukturiert Speichern
- Datensätze sind wiederum sehr klein
- Echtzeitabfrage von indizierten Datensätzen
- einfache Schnittstelle (PUT, GET, SELECT, ...)
- kurz: Eine Datenbank in der Cloud
- Ein Kommentar: „*Amazon SimpleDB is not a database!*“ [NS07]

Bild - Das Datenmodell als Tabelle



	A	B	C	D	E	F	G	H
1		**Attribute 1**	**Attribute 2**	**Attribute 3**	**...**	**Attribute <n>**		
2	**Item 1**	value	value	value	value	value		
3	**Item 2**	value	value	value	value	value		
4	**Item 3**	value	value	value	value	value		
5	**...**	value	value	value	value	value		
6	**Item <n>**	value	value	value	value	value		
7								
8								
9								
10								
11								
12								

Domain 1 | Domain 2 | Domain 3 | ... | **Domain <n>**

Abbildung: Datenmodell als Tabelle

Was ist Amazon SimpleDB? (2)

- Datensätze sind in „domains“ gruppiert
- keinerlei Beziehungen zwischen „domains“ möglich
- Datensätze sind „items“ und enthalten „attributes“
- ein „attribute“ kann mehrere Werte enthalten
- „items“ und „attributes“ haben eindeutige Namen
- die Kombination aus beiden legt fest auf welchen Wert man zugreift
- die Namen sowie die Werte sind Strings [1]

[1] UTF-8 kodiert, alle Zeichen möglich die in XML erlaubt sind

Ein kurzes Beispiel

Programmierbeispiel in Python

```
import boto

sdb = boto.connect_sdb("ACCESS_KEY","SECRET_ACCESS_KEY")
domain = sdb.get_domain("test")

for item in domain:
    print item.name
```

1. Die Authentifizierung mit `ACCESS_KEY` und `SECRET_ACCESS_KEY` wird gespeichert in „sdb“
2. ein Handle der „domain“ „test“ wird in `\domain\` gespeichert
3. die for-Schleife gibt die Namen aller Datensätze innerhalb der Domain „test“ aus

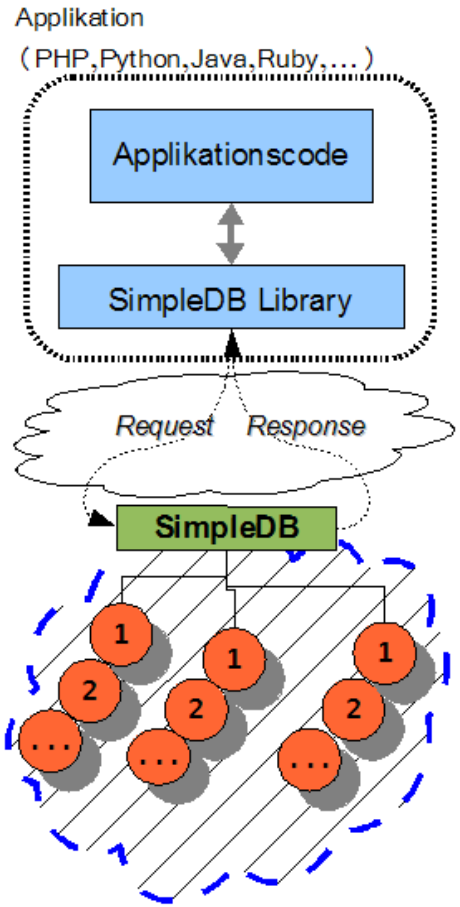
Wichtig: Alle Anfragen an den SimpleDB Service laufen im Hintergrund ab!

Was will Amazon mit SimpleDB abdecken?

- AWS hat einige selbstgesteckte Ziele mit SimpleDB
- 80% der Anforderung an Datenbanken abdecken
- Verfügbarkeit, Stabilität und Skalierbarkeit
- sofortige Konsistenz wird dafür geopfert

Die Konzepte dahinter

- verteilter Datenspeicher
- Stichwort: „Eventual Consistency“ [CY07]
- Datenmodell: **kurz:** eine riesige Tabelle, (ähnlich Google's BigTable Konzepten . . .)
- REST(ful)[a] oder SOAP Aufrufe über https
- nur SQL ähnliche Funktionalität mit SELECT (keine JOINs)
- kein Schema, unterschiedliche Anzahl von „attributes“ mit unterschiedlichen Namen in einer „domain“
- keine Typen nur Strings, dadurch: lexikographische Ordnung



[a] „*I guess it is RESTful, but Damn is it ugly.*“ [DO07]

Die Rolle innerhalb der Amazon Web-Services

- einfache Zusammenarbeit mit EC2 und S3
- Transfer innerhalb der AWS kostenlos
- eine Anwendung läuft zum Beispiel auf EC2
- Speicherung der Daten in S3
- Meta-Daten und Keys in SimpleDB (verweisen auf Objekte in S3)
- der Vorteil: schnelle Suche auf den Meta-Daten

Vorteile (1)

- Einfache Schnittstellen
- sofortige Verfügbarkeit, keine eigenen Datenbanksysteme notwendig
- spontane Skalierung (bis 10 GB pro „domain“, bei 100 „domains“)
- eine verteilte Datenbank, dadurch Stabilität und Parallelität
- hoher Durchsatz bei vielen gleichzeitigen Anfragen,
  Skalierung mit der Last

Vorteile (2)

- keine Fixkosten, man bezahlt nur für aktuellen Bedarf
- kein festes Schema, Datensätze innerhalb der selben „domain“ können unterschiedliche Anzahl von Attributen besitzen
- innerhalb der AWS geringe Latenz ($\tilde{5}$ms)
  dadurch hohe Geschwindigkeit bei Anwendungen innerhalb der AWS
- keine replizieren von Daten nötig, Daten sind durch das verteilen automatisch an verschiedenen Orten
- Backups sind automatisch

Nachteile (1)

- *keine* Unterstützung für *Service Level Agreements (SLA)*
- keine direkte Kontrolle über die Datenbank und die Art der Speicherung (betrifft Geschwindigkeit)
- Daten liegen bei einem Dritten (Stichwort: Datenschutz)
- keine relationale Datenbank (keine JOINs)
- nicht kompatibel mit Datenbankstandards (`SQL`)
- keine Typen, dadurch anfälliger für Programmierfehler

Nachteile (2)

- nur **lexikographische Ordnung**
  es ensteht mehr Aufwand in der Implemenntierung für Zahlen (z.B.: 00012345, „Offset“ für negative Werte, Datum/Zeiten konvertieren in ISO-8601)
- Konsistenz nicht sofort
  direkte Abrufe nach Speichern lieferen möglicherweise noch alte Daten
- serielle Anfragen durch Latenzzeiten verlangsamt
  hohe Parallelität der Anwendungen für hohen Durchsatz erforderlich
- sowie die folgenden technischen Beschränkungen . . .

Die Grenzen ...

Domain size	10 GB per domain
Domain size	1 billion attributes per domain
Domain name	3-255 characters (a-z, A-Z, 0-9, '_', '-', and '.')
Domains per account	100
Attribute name-value pairs per item	256
Attribute name length	1024 bytes
Attribute value length	1024 bytes
Item name length	1024 bytes
Attribute name, attribute value, and item name allowed characters	All UTF-8 characters that are valid in XML documents.Control characters and any sequences that are not valid in XML are not allowed.
Attributes per PutAttributes operation	256
Attributes requested per Select or QueryWithAttributes operation	256
Maximum items in query response	250
Maximum query execution time	5 seconds
Maximum predicates per query expression	10
Maximum comparisons per query expression predicate	10
Maximum number of unique attributes per select expression	20
Maximum number of comparisons per select expression	20
Maximum response size for QueryWithAttributes and Select	1MB

Bibliotheken in ...

- C, C#, Groovy, Java, Perl, PHP, Python, Ruby, Scala, Visual Basic.net, ...
- Der Großteil wurde nicht von Amazon entwickelt
- keine (guten) Tools zur schnellen Bearbeitung von Datensätzen
- eindeutiger Fokus auf Entwickler

„SimplePoll“ - Implementierungsbeispiel

- „SimplePoll“, versuch einer Web-Applikation in Python mit SimpleDB
- das genutzte Framework ist „CherryPy“, sowie „boto“[2]
- kein Anspruch auf Vollständigkeit und Fehlerlosigkeit ;-)
- Ziel war eine minimale Anwendung für Abstimmungen
- Demonstration aller Grundlegenden SimpleDB Funktionen

[2] [BOTO] [CHEPY]

Kosten (1)

- Berechnungszeit
  - Die ersten 25 Stunden CPU-Zeit pro Monat sind kostenlos[3]
  - $0.14 für jede weitere Stunde CPU-Zeit
  - Leicht zu überprüfen, jede Anfrage gibt die verbrauchte CPU-Zeit zurück
- Speicher
  - 1 GB Speicher pro Monat ist kostenlos
  - $0.25 pro GB/Monat danach
- berechnet wird die größe der Daten + 45 bytes pro “item“,“attribute“ und “attribute-value“-Paar

[3] gemessen an einer Stunde CPU-Zeit einer 1.7 Ghz Xeon CPU von 2007

Kosten (2)

- Traffic (eingehend)
  - 1 GB Daten pro Monat kostenlos
  - $0.10 pro GB - eingehender Traffic
- Traffic (ausgehend)
  - 1 GB Daten pro Monat kostenlos
  - $0.170 pro GB für die ersten 10 TB im Monat
  - $0.130 pro GB - bei über 40 TB im Monat
  - $0.110 pro GB - bei über 100 TB im Monat
  - $0.10 pro GB - bei über 150 TB im Monat
- **Wichtig:** Transfer innerhalb der AWS der selben Region kostenlos!

Einsatzgebiete - Wer könnte profitieren?

- Einsatz z.B. als Logging Dienst für eine komplexe Web-Applikation (Videokonvertierung)
- Kataloge, E-Commerce
- Verzeichnisdienste
- Benutzerverwaltung, Community-Verwaltung („Social Networks“)
- Am Wichtigsten: Die Anforderung an die Datenbank möglichst genau spezifieren
- Testen: Amazon bietet ein gewisses kostenloses Kontingent

API Überblick - Version 2009-14-05

- Requests RESTful oder SOAP gekapselt
- alle Anfragen signiert (mit AWS Secret Access Key)
- seit dieser Version nur noch per https
- folgende Operationen in dieser Version
- Operationen auf „domains“:
  `CreateDomain, DeleteDomain, ListDomains, DomainMetadata`
- Operationen auf Datensätzen:
  `PutAttributes, GetAttributes, DeleteAttributes`
  `Select, BatchPutAttributes`
- *Alle Operation liefern die benötigte CPU-Zeit sowie eine ID der Anfrage zurück*
- die folgenden Folien präsentieren kurz die Operationen als REST-Anfragen

API(1) - CreateDomain, DeleteDomain, ListDomains

CreateDomain

BENÖTIGTE PARAMETER:

`DomainName` - Name der zu erstellenden „domain“

Erstellt die „domain“ `DomainName` , der Name muss einzigartig in einem AWS Konto sein.

DeleteDomain

BENÖTIGTE PARAMETER:

`DomainName` - Name der zu löschenden „domain“

Löscht „domain“ `DomainName`, inlusive aller Datensätze.

ListDomain

OPTIONALE PARAMETER:

`MaxNumberOfDomains` - maximale Anzahl der Namen die zurückggeben werden (1-100, Vorgabe: 100)

`NextToken` - String der den Anfang der folgenden Liste von Namen angibt

Gibt alle „domains“ dieses AWS Kontos zurück, bis zu `MaxNumberOfDomains`. Ein `NextToken` wird zurückggeben wenn es mehr als `MaxNumberOfDomains` gibt. Ein nachfolgender Aufruf mit diesem `NextToken` gibt die nächsten Namen zurück bis zu `MaxNumberOfDomains`, und so weiter.

API(2) - DomainMetaData, PutAttributes

DomainMetaData

BENÖTIGTE PARAMETER:

`DomainName` - Name der „domain“, für welche meta-Daten angefordert werden

Gibt Informationen über die „domain“ `DomainName` zurück, inklusive den Zeitpunkt der Erstellung, Anzahl der „items“ und „attributes“, sowie die Länge von „attribute“-Namen und Werten.

PutAttributes

BENÖTIGTE PARAMETER:

`Attribute.X.Name` - Name des „attributes“ X (X ist die Nummer des „attributes“, 0 bis 255)
`Attribute.X.Value` - Wert des „attributes“ X
`ItemName` - Name des zu ändernden „items“
`DomainName` - Name der Domain in der der Eintrag gespeichert wird

OPTIONALE PARAMETER:

`Attribute.X.Replace` - `true` oder `false`, überschreibe „attribute“/“value“-Paare (Vorgabe `false`)

Erstellt oder ersetzt „attributes“ eines Datensatzes, durch das Paar von „attribute“-Name und Wert werden diese eindeutig festgelegt. Das erste „attribute“ wird mit `Atttribute.0.Name` bezeichnet und `Attribute.0.Value` ist dessen Wert. Wenn `Attribute.X.Replace`, „true“ ist, werden alle „attributes“ mit dem Namen `Attribute.X.Name` mit dem angegebenen Wert überschrieben.

API(3) - GetAttributes, DeleteAttributes

GetAttributes

BENÖTIGTE PARAMETER:

`ItemName` - Name des Datensatzes

`DomainName` - Name der „domain“, in welcher der Datensatz liegt

OPTIONALE PARAMETER:

`AttributeName.X` - Ein oder mehrere „attribute“-Namen, die zurückgegeben werden sollen

Gibt alle „attributes“ eines Datensatzes aus, oder nur die, die in der Anfrage angegeben sind. Gibt keinen Fehler aus falls der Datensatz nicht existiert, da nicht garantiert werden kann, dass er nicht in einer anderen Instanz vorhanden ist.

DeleteAttributes

BENÖTIGTE PARAMETER:

`ItemName` - Name des „items“

`DomainName` - Name der Domain in der das „item“ gelöscht werden soll

OPTIONALE PARAMETER:

`Attribute.X.Name` - Name des zu löschenden „attributes“ X (X ist die Nummer des „attributes“, 0 bis 255)

`Attribute.X.Value` - Wert des zu löschenden „attributes“ X, falls es mehrere Werte hat

Löscht eines oder mehrere „attributes“ des „items“ `ItemName`. Falls alle „attributes“ gelöscht wurden, wird auch das `ItemName` gelöscht.

API(4) - Select, BatchPutAttributes

Select

BENÖTIGTE PARAMETER:

`SelectExpression` - der Ausdruck der zur Abfrage benutzt wird

OPTIONALE PARAMETER:

`NextToken` - gibt den Anfang der folgenden Liste von „item“-Namen an

Gibt einen Satz von `Attributes` von `ItemNames` zurück die dem Audruck entsprechen. Ähnlich eines SQL `SELECT` Ausdrucks. Die Gesamtgröße der Antwort darf 1 MB nicht überschreiten, es werden nur soviele „items“ zurückggeben, dass die Antwort nicht größer wird. Die verbleidenden „items“ werden mit Angabe des `NextToken` abgefragt.

BatchPutAttributes

BENÖTIGTE PARAMETER:

`DomainName` - Name der Domain in der das „item“ gesucht gelöscht werden soll
`Item.Y.ItemName` - Name des „item“ Y (Y gibt die Nummer innerhalb der Anfrage an)
`Item.Y.Attribute.X.Name` - Name des „attribute“ X des „items“ Y
`Item.Y.Attribute.X.Value` - Wert des „attribute“ X des „items“ Y

OPTIONALE PARAMETER:

`Item.Y.Attribute.X.Replace` - `true` oder `false` (Vorgabe `false`)

Kombiniert einzelne `PutAttributes`-Operationen in einer, zur verringerung der Latenz und Optimierung des Durchsatzes. Die Parameter haben die selbe Bedeutung wie bei `PutAttributes`.
`Item.0.ItemName` gibt das erste zu schreibende „item“ in der Anfrage an.
`Item.0.Attribute.0.Name` und `Item.0.Attribute.0.Name` das erste zu schreibende „attribute“/“value“-Paar.

Zusammenfassung

- Amazon SimpleDB übernimmt typische Datenbank Aufgaben, wenn man mit den Beschränkungen umgehen kann
- für aktuelle Web-Anwendungen geeignet
- Nutzen hängt extrem vom Einsatzgebiet ab (keine allround-Lösung)
- die Komplexität einer Implementierung nicht sofort Ersichtlich (z.B. bei Ordnung auf Zahlen)
- aber auch einfache Programmierung bei Standardaufgaben
- gute Skalierung, aber nicht unbegrenzt (10 GB pro „domain“)
- Geschwindikeit Abhängig von der Parallelität der Anwendung
- noch als öffentliche Beta deklariet, d.h. Schnittstellen ändern sich regelmässig aber auch Verbesserungen am Service
- noch keine Unterstützung für Service Level Agreements (SLA)
- im Fazit: trotzallem eine gute Ergänzung zum bestehenden Angebot innerhalb der Amazon Web Services

Abschluss mit einem Zitat

„Depending on how you weight each factor, SimpleDB could be way behind or way ahead of other options. What's interesting is to see what people think is important. For many people the only real database is relational and if it doesn't have transactions, joins, etc it's not real. Databases like beauty seem to be in the eye of the beholder.“ [HS07]

Quellen (1)


N. Shalom, *Amazon SimpleDB is not a database!*, 30.12.2007
http://natishalom.typepad.com/nati_shaloms_blog/2007/12/amazon-simpledb.html
(Abruf am 21.05.09)


C. H. Ying, *What You Need To Know About Amazon SimpleDB*, 13.12.2007
http://www.satine.org/archives/2007/12/13/amazon-simpledb/
(Abruf am 17.05.09)


T. Hoff, *The Current Pros and Cons List for SimpleDB*, 15.12.2007
http://highscalability.com/current-pros-and-cons-list-simpledb
(Abruf am 17.05.09)


D. Obasanjo, *Amazon SimpleDB: The Good, the Bad and the Ugly*, 21.12.2007
http://www.25hoursaday.com/weblog/2007/12/21/
AmazonSimpleDBTheGoodTheBadAndTheUgly.aspx
(Abruf am 17.05.09)

Quellen (2)


AWS, *Amazon SimpleDB Getting Started Guide (API Version 2009-04-15)*, 21.05.2009
`http://docs.amazonwebservices.com/AmazonSimpleDB/2009-04-15/GettingStartedGuide/`
(Abruf am 23.05.09)


AWS, *Amazon SimpleDB Developer Guide (API Version 2009-04-15)*, 21.05.2009
`http://docs.amazonwebservices.com/AmazonSimpleDB/2009-04-15/DeveloperGuide/\`
(Abruf am 23.05.09)


„boto, Python interface to Amazon Web Services“
`http://code.google.com/p/boto/` (Abruf am 21.05.09)


„CherryPy, object-oriented HTTP framework“
`http://www.cherrypy.org/` (Abruf am 24.05.09)


M. Garnaat, *Introduction to SimpleDB in Boto*, 28.12.07
`http://code.google.com/p/boto/wiki/SimpleDbIntro` (Abruf am 21.05.09)