OPITZ CONSULTING

Menschen. Innovationen. Lösungen.

Polyglot Persistence und NoSQL

Mehr Flexibilität, mehr Komplexität!?

Stefan Kühnlein
Solution Architekt

OPITZ CONSULTING GmbH

2014 DOAG Konferenz + Ausstellung 18. - 20. November | Nürnberg

Nürnberg, 18.11.2014

Agenda

1. Rückblick – Relationale Datenbanken
2. NoSQL-Datenbanken
3. Entwicklung von NoSQL-Anwendungen
4. Polyglot Persistence

1 Rückblick – Relationale Datenbanken

Relationale Datenbank

- **Relationale Datenbank bevorzugt für die Speicherung von Daten**
  - es existieren ausgereifte Lösungen
  - breite Akzeptanz und Vertrautheit
  - unzählige Tools
- **Speicherung der Daten in**
  - 2 dimensionalen Tabellen
  - Verfügt über ein klar definiertes Schema inkl. Relationen und Constraints
- **Unterstützung der SQL (Structured Query Language)**
  - eine „standardisierte“ Abfragesprache
  - sehr flexible
  - viele Operationen
- **Relationale Datenbank haben sich zum Standard entwickelt**

Kennzeichen von Relationalen Datenbanken

- **Transaktional**
  - ACID
    - Automicity
    - Consistency
    - Isolation
    - Durability
- **Architekturentscheidungen**
  - Jeder End-User interagiert direkt mit der Datenbank
    - RDBMS verwaltet die Integrität und Concurrency
    - Unterstützung von unbekannten Anwendungspattern
      - Flexible Abfragesprache
      - Datenstrukturen ohne Ausrichtung auf eine bestimmtes Abfragemuster
  - Datenbank läuft auf einen einzigen Rechner
    - Sicherstellung des ACID-Anforderungen

Grenzen der Relationalen Datenbanken

- **Entwicklungen von immer komplexeren Anwendungen auf Basis von Relationalen Datenbanken**
  - Auslagerung der Businesslogik in eine separate Schicht
  - Die Entwicklung von Anwendung erfolgt mehr über einen objektorientierten Ansatz statt einem prozeduralen Ansatz
  - Einsatz von speziellen Frameworks
- **Skalierbarkeit**
  - Datenbanken wachsen stetig und geraten außer Kontrolle
  - Performance verschlechtert sich exponentiell

Empfehlung 1 – Optimierung von Relationalen Datenbanken

- **Indizierung der „ganzen“ Datenbank**
  -> geringfügig schnellere Ausführung von Datenbankabfrage
- **Erstellung von Materialized Views für komplexe Joins**
  -> Albtraum diese zu managen
  -> veralten ständig
- **Denormalisierung**
  -> Redundanzen
- **Einführung von Caches**
  -> Daten veralten und Cache muss konsistent gehalten werden
  -> Weitere Redundanzen

Empfehlung 2 – Skalierung von Relationalen Datenbanken

- **Einführung von Master/Slave**
  - Nur möglich unter der Annahme dass mehr lesend auf die Daten zugegriffen werden
  - Schreiben auf dem Master; Lesen von den Slaves
  - Der Master muss seine Daten zu den Slaves replizieren
  - Stellt diese Lösung konsistente Daten zur Verfügung?

- **Sharding**
  - Verbessert sowohl lesende als auch schreibende Zugriffe
  - Kein Join über Partitionen
  - Keine Referenzielle Integrität
  - Benötigt eine Anpassung der Anwendungen
  - Einführung eines Single-Point of Failures
  - Stellt diese Lösung konsistente Daten zur Verfügung?

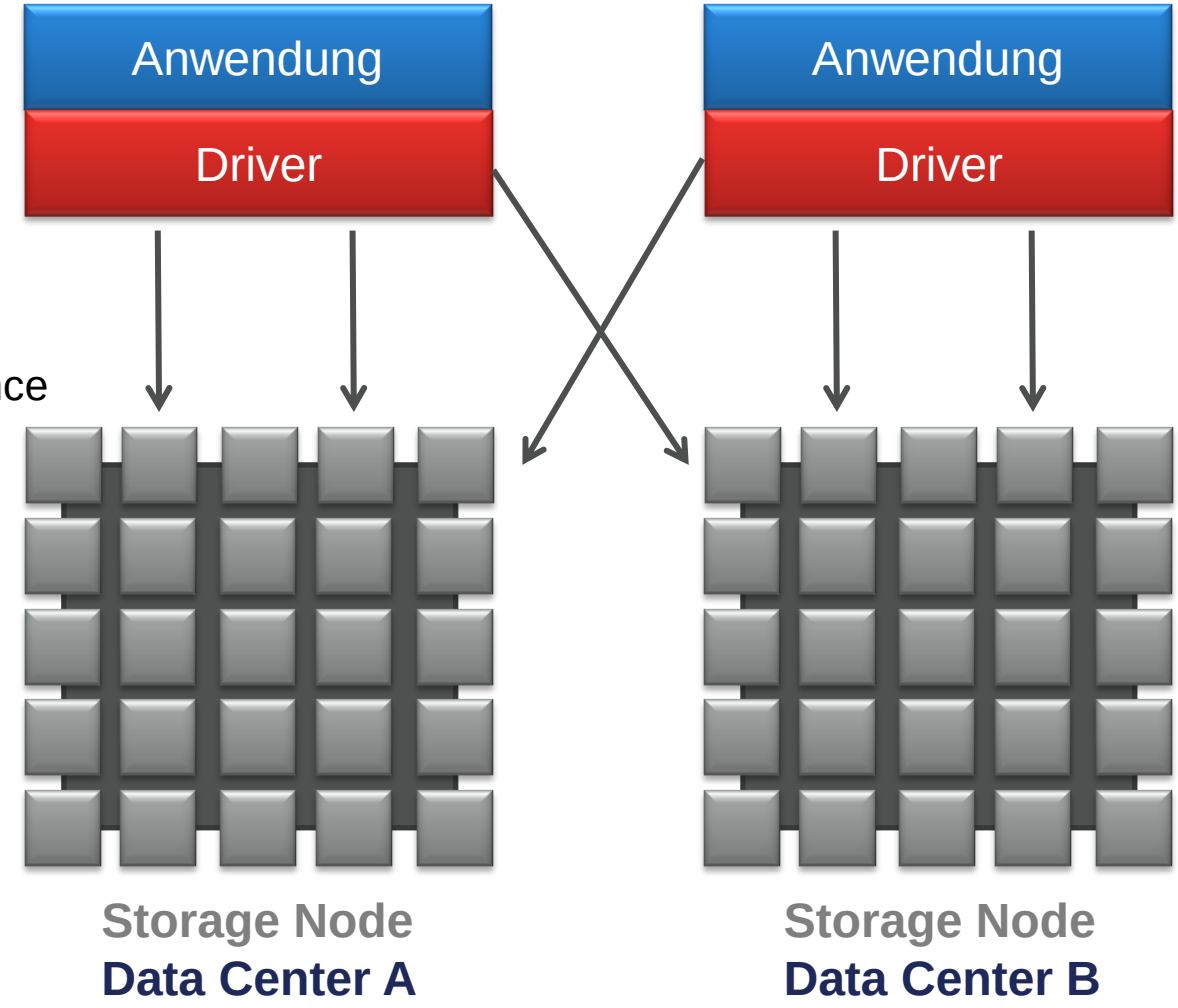
2 NoSQL-Datenbanken

BASE Akronym

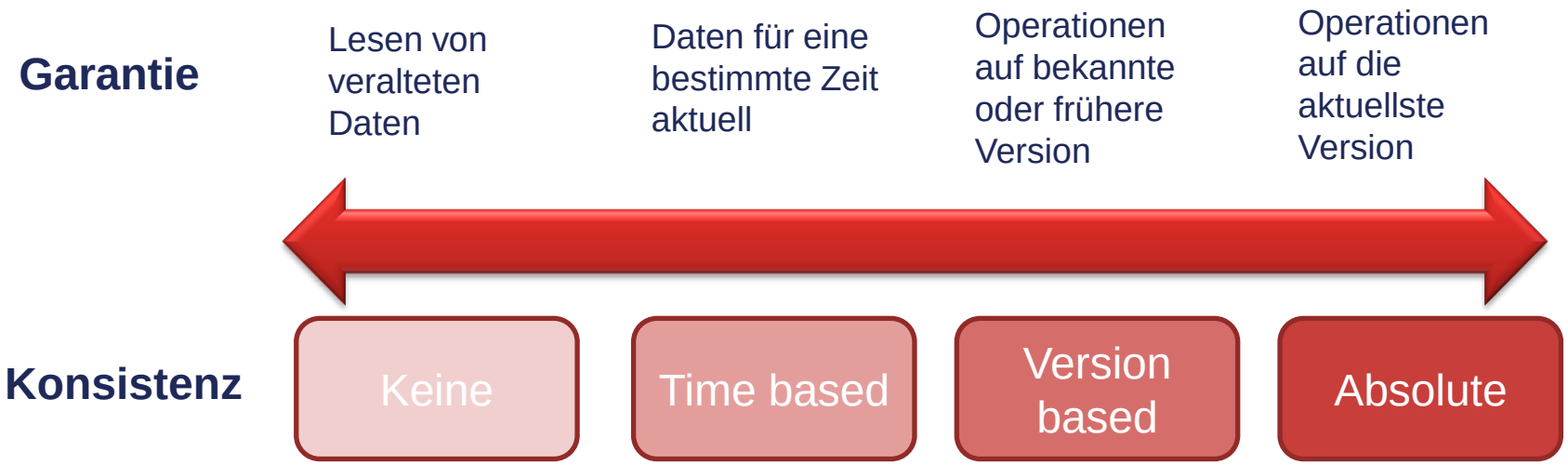
- **Basically Available**
  - Verteilung der Daten durch Replikation auf verschiedene Partitionen
  - Zugriff auf Daten möglich auch wenn nicht alle Konten verfügbar sind
- **Soft State**
  - ACID-Systeme gewährleisten die Konsistenz zu jedem beliebigen Zeitpunkt
  - Bei NoSQL-Datenbanken wird den Entwickler überlassen, wie damit umgegangen werden soll
- **Eventually consistent**
  - Daten werden erst zu einem späteren Zeitpunkt wieder konsistent

Oracle NoSQL Datenbank

- Einfache Administration
- Flexibles Datenmodell
- Voraussagbare Performance
- ACID Transaktionen
- Transparente Integration



Isolation Level in Oracle NoSQL DB



SQL vs. NoSQL

Zentralisiert auf eine Maschine	Verteilt auf einen Cluster
CA	AP/CA/CP
Vertikale Skalierung	Horizontale Saklierung
SQL	Spezifische APIs
ACID	BASE
Index für jedes Attribut	Typischerweise auf Keys
Flexible Abfragen	Vordefinierte Abfragen

3 Entwicklung von NoSQL- Anwendungen

CRUD Operationen für NoSQL

- **Erstellen bzw. Aktualisieren von neuen Datensätzen**
  - put
  - putIfAbsent
  - putIfPresent
  - putIfVersion
- **Löschen von Datensätzen**
  - delete
  - deleteIfVersion
- **Lesen von Datensätzen**
  - get
  - multiGet
- **Iterators**
  - multiGetIterator
  - storeIterator

EclipseLink und NoSQL

- **Support für JPA-style Access**
- **Definition von Annotationen und XML zur Identifizierung von NoSQL Entitäten**
- **Support von JPQL**
- **Initialer Implementierung für Oracle NoSQL und MongoDB**
- **Unterstütz den Mix von relationalen und nicht relationalen Daten in eine einzigen Persistence-Unit**
- **Grundlegende JPA-Konzepte auf NoSQL Datenbanken anwendbar**
  - Persistence Entity, Embeddables, ElementCollection, OneToOne,…

NoSQL und JPA mit EclipseLink

- **Speicherung von JSON-Objekten mit JPA in 6 Schritten**

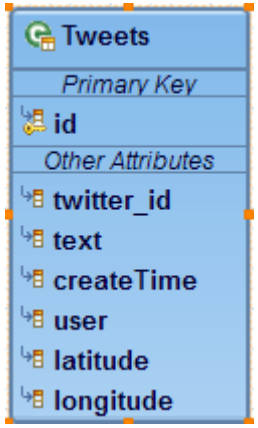
1. Mapping der Objekte

```
@Entity
@NoSql(dataFormat=DataFormatType.MAPPED)
public class Tweet
```

2. Definition der IDs

```
@Id
@GeneratedValue
private String id;
```

3. Mapping der Attribute



NoSQL und JPA mit EclipseLink

4. Definition der Lockingstrategie

```
@Version
private long version;
```

5. Definition der Abfragen

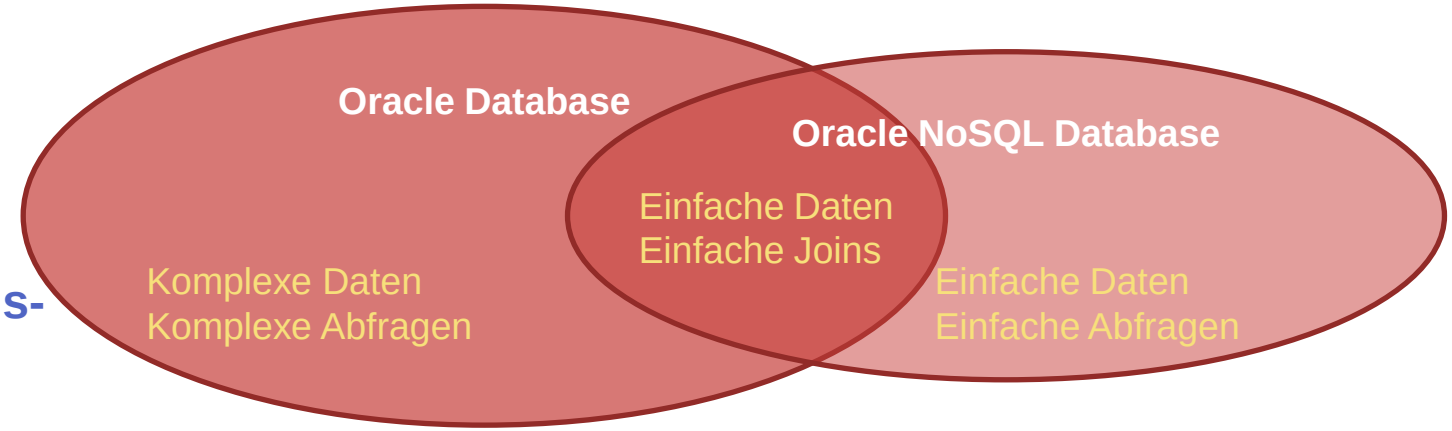
```
@NamedQuery / @ NamedNatvieQuery
        oder
Query query = em.createQuery("Select o from Tweet o where text like ,‘%DOAG2014%‘);
List<Tweets> tweets = query.getResultList();
```

6. Spezifikation der Verbindung zur Datenbank

```
<persistence-unit name="ondb" transaction-type="RESOURCE_LOCAL">
<class>de.oc.model.Tweet</class>
<properties> <property name="eclipselink.target-database"
        value="org.eclipse.persistence.nosql.adapters.nosql.OracleNoSQLPlatform"/>
<property name="eclipselink.nosql.connection-spec"
        value="org.eclipse.persistence.nosql.adapters.nosql.OracleNoSQLConnectionSpec"/>
<property name="eclipselink.nosql.property.nosql.host" value="Localhost:5000"/>
<property name="eclipselink.nosql.property.nosql.store" value="kvstore"/>
<property name="eclipselink.logging.level" value="FINEST"/>
```

4 Polyglot Persistence

Relationale Datenbanken vs. NoSQL



Applikations-typ

Applikation
Data Management
Charakteristiken

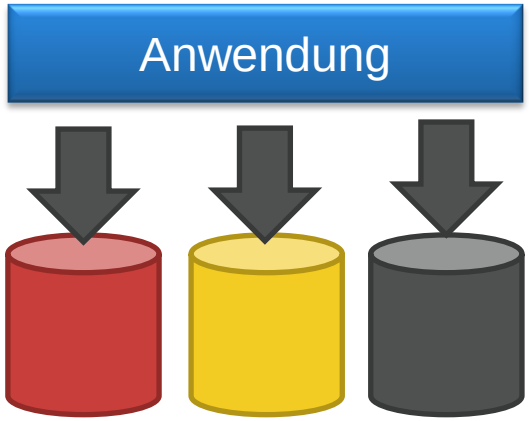
Voller Funktionsumfang
Zentral verwaltete Daten
Hohe Performance
Verwaltung durch DBA
Allgemein verwendbar

Low TOC
TBs von verteilten Daten
Niedrige Latenzzeiten
Flexibles Schema
Spezielle Verwendung

Polyglot Persistence

- **Enterprise werden immer komplexer**
  - Die Speicherung der Daten in einer einzigen Datenbank ist absurd
  - Die Speicherung der Daten in einem einzigen Modell führt sicherlich zu weiteren Problemen

- **Polyglot Persistence**
  - Verwendung von multiplen Datenspeichern, basierend auf der Art und Weise, wie die Anwendung die Daten benötigt
    - Relationale Daten für die Speicherung von Benutzerinformationen, Rechnungen…
    - Dokumentenorientierte NoSQL-Datenbanken für die Speicherung von Tweets, HitCounts, Logs…
    - Graphorientierte NoSQL-Datenbanken für die Speicherung von Profilen aus den sozialen Netzwerken



Herausforderungen

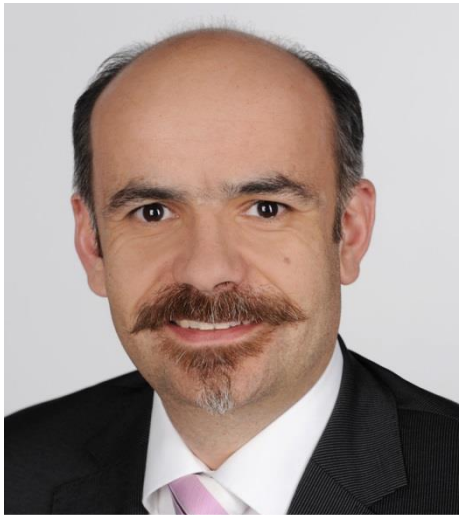
- **Architektur**
  - Steigerung der Komplexität der gesamten Anwendungslandschaft
- **Implementierung**
  - Unterschiedliche Abfragesprache
  - Unterschiedliche Datenstrukturen
- **Betrieb**
  - neue Ausausforderungen bzgl. separater
  - verschiedene Konzepte bzgl. Hochverfügbarkeit

Fragen und Antworten

Kontakt

Stefan Kühnlein
Solution Architect

OPITZ CONSULTING GmbH
Weltenburger Str. 4| 81677 München
Tel. +49 (89) 680098-0
stefan.kuehnlein@opitz-consulting.com



youtube.com/opitzconsulting

@OC_WIRE

slideshare.net/opitzconsulting

xing.com/net/opitzconsulting