JAVAFORUM stuttgart

adesso business. people. technology.

NoSQL für Java-Entwickler

Java Forum Stuttgart 2013

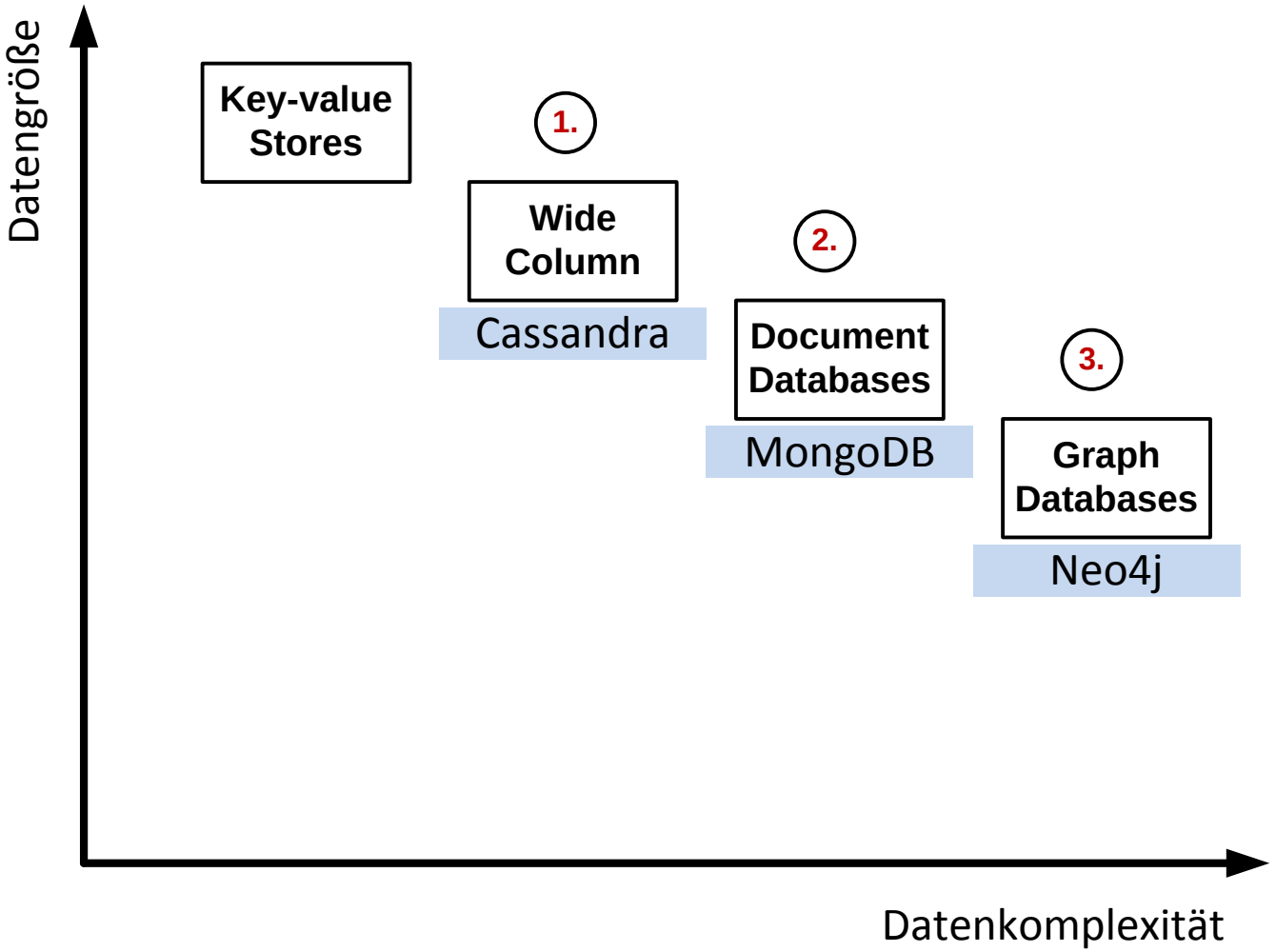
Kai.Spichale@adesso.de

twitter.com/kspichale

spichale.blogspot.de

23.06.2013

Agenda


Datengröße
Key-value Stores
1.
Wide Column
Cassandra
2.
Document Databases
MongoDB
3.
Graph Databases
Neo4j
Datenkomplexität

Apache Cassandra

http://cassandra.apache.org/

Apache Cassandra

Beschreibung

- Verteilter Wide Column Store
- Kombiniert Features von Amazon Dynamo und Google Big Table
- Apache License 2.0
- Java

Wichtige Features

- Horizontale Skalierung durch Sharding
- Replikation (wählbare Konsistenzstufen)
- High Performance I/O
- Fehlertolerant

Kommerzieller Support

- DataStax

Hector Client API

```
Cluster cluster = HFactory.getOrCreateCluster("cluster", "localhost:9160");
Keyspace keyspace = HFactory.createKeyspace("demo", cluster);

ColumnFamilyTemplate<String, String> template
  = new ThriftColumnFamilyTemplate<String, String>(
                              keyspace, "user",
                              StringSerializer.get(), StringSerializer.get());


Stiring id = TimeUUIDUtils.getTimeUUID(new Date().getTime()).toString();

ColumnFamilyUpdater<String, String> updater = template.createUpdater(id);

updater.setString(FIRSTNAME, "Kai");
updater.setString(LASTNAME, "Spichale");

template.update(updater);
```

Hector Client API

Eigenschaften:

- Clientseitiges Failover
- Connection Pooling
- Load Balancing

- Kapselt Cassandra Thrift API
- Einfacher Object Mapper
  - Keine Unterstützung für CQL 3 Features wie Compound Keys

JDBC Driver für Cassandra

```
Connection con = DriverManager.getConnection(
        "jdbc:cassandra://102.5.12.121:9160/system?version=3.0.0");

String query = "CREATE TABLE tweet (
                                    user_id varchar,
                                    tweet_id varchar,
                                    author varchar,
                                    message varchar,
                                    PRIMARY KEY(user_id,tweet_id)
                                  );";

PreparedStatement statement = con.prepareStatement(query);

statement.execute();
statement.close();
```

Beispiel: Twitter timeline

Logische Sicht auf Tabelle timeline

user_id	tweet_id	author	message
Kai	4811	Bob	I love coffee
Kai	0815	Paul	Thank god it‘s friday
Mike	4811	Bob	I love coffee

```
CREATE TABLE timeline (
    user_id varchar,
    tweet_id uuid,
    author varchar,
    message varchar,
    PRIMARY KEY (user_id, tweet_id)
);
```

Beispiel: Twitter timeline

Logische Sicht auf Tabelle timeline

user_id	tweet_id	author	message
Kai	4811	Bob	I love coffee
Kai	0815	Paul	Thank god it‘s friday
Mike	4811	Bob	I love coffee

Physisches Layout mit Composite Columns

Kai	[4811,author]: Bob	[4811, message]: I love coffee	[0815,author]: Paul	[0815,body]: Thank god it‘s friday
Mike	[4811,author]: Bob	[4811, body]: I love coffee		

Kundera: Object Mapping

```
@Entity
@Table(name="tweet", schema="demo@cassandra_pu")
public class TweetEntity {

  @EmbeddedId
  private TweetEntityPK pk;

  @Column
  private String author;

  @Column
  private String message;

  // ...
}
```

Kundera: Zusammengesetzter Schlüssel

```java
@Embeddable
public class TweetEntityPK {

  @Column(name = "tweet_id")
  private String tweetId;

  @Column(name = "user_id")
  private String userId;

  // ...

}
```

Kundera: JPA-ähnliche Persistenz

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("pu");
EntityManager em = emf.createEntityManager();
em.setProperty("cql.version", "3.0.0");


weetEntityPK pk = new TweetEntityPK();
pk.setUserId("42");
pk.setTweetId(UUID.randomUUID().toString());


TweetEntity tweet = new TweetEntity();
tweet.setPk(pk);
tweet.setAuthor("Kai");
tweet.setMessage("Java Forum Stuttgart 2013");


em.persist(tweet);
```

Kundera: JPA-ähnliche Persistenz

```
TweetEntity tweet = em.find(TweetEntity.class, pk);


Query q = em.createNativeQuery(
  "select * from tweet where user_id = '42' limit 10;", TweetEntity.class);


List<TweetEntity> list = q.getResultList();
```

MongoDB

http://www.mongodb.org/

MongoDB

Beschreibung

- JSON-Dokumentendatenbank
- GNU AGPL v3.0
- Implementiert in C++

Wichtige Features

- Horizontale Skalierung durch Sharding
- Master-Slave-Replikation
- Server-seitige JavaScript-Ausführung

Entwicklung und kommerzieller Support

- 10gen

Mongo Java Driver

```
MongoClient client = new MongoClient(new ServerAddress("localhost", 27017));
DB database = client.getDB("demo");
DBCollection collection = database.getCollection("events");

BasicDBObject event = new BasicDBObject();
event.append("title", "Java Forum Stuttgart 2013");
event.append("participant_number", 1500);

BasicDBObject venue = new BasicDBObject();
venue.append("city", "Stuttgart");
event.append("venue", venue);

collection.insert(event);
```

Mongo Java Driver

```
AggregationOutput output = collection.aggregate(
  // Group events
  new BasicDBObject("$group",
    new BasicDBObject("_id", "$venue.city")
    .append("count", new BasicDBObject("$sum","$participant_number"))),

  // Sort grouped elements
  new BasicDBObject("$sort",
    new BasicDBObject("count", 1))

  );


for (DBObject doc : output.results()) {
  String city = (String) doc.get("city");
  Double count = (Double) doc.get("count");

  // ...
}
```

Spring Data

Beschreibung

- Entwickelt von VMware / SpringSource
- Apache License v2.0
- Unterstützung: MongoDB, Neo4j, Redis, Hbase, Apache Hadoop, JPA, …

Wichtige Features

- Repository Support
  - Abfragen abgeleitet von Methodensignaturen
- Object / Document Mapping
  - Mittels Annotationen wie @Document, @NodeEntity, @Indexed, etc.
- Templating
  - Konfiguration von Verbindungen
  - Abstraktion von Ressourcen

Neo4j

http://www.neo4j.org/

Neo4j

Beschreibung

- Graphenorientierte Datenbank
- AGPLv3 / kommerziell
- Implementiert in Java

Wichtige Features

- ACID-Transaktionen
- Traversal Framework für schnelle Graphenabfragen
- Disk-basiert: Spezielles binäres Festplattenformat

Entwicklung und kommerzieller Support

- Neo Technology

Neo4j Spring Data: Einen Graphen speichern

```
GraphDatabaseService graphDb = new
  GraphDatabaseFactory().newEmbeddedDatabase(DB_PATH);

Transaction tx = graphDb.beginTx();

try {
 Node bob = graphDb.createNode();
 bob.setProperty("name", "Bob");

 Node alex = graphDb.createNode();
 alex.setProperty("name", "Alex");

 bob.createRelationshipTo(alex, DynamicRelationshipType.withName("FOLLOWS"));

 tx.success();

} finally {
 tx.finish();
}
```

Neo4j Spring Data: Der kürzeste Weg

```
Node bob = indexService.get("name", "Bob").getSingle();
Node alex = indexService.get("name", "Alex").getSingle();


DynamicRelationshipType rel = DynamicRelationshipType.withName("FOLLOWS");


PathFinder<Path> finder = GraphAlgoFactory.shortestPath(
  Traversal.expanderForTypes(rel, Direction.BOTH), MAX_DEPTH);

Path path = finder.findSinglePath(bob, alex);
```

Neo4j Spring Data: Anzahl verbundener Benutzer

```
TraversalDescription td = Traversal
                          .description()
                          .breadthFirst()
                          .relationships(FOLLOWS, Direction.OUTGOING)
                          .evaluator(Evaluators.excludeStartPosition())
                          .evaluator(Evaluators.toDepth(DEPTH));

Traverser friendsTraverser = td.traverse(node);

int[] counters = new int[DEPTH+1];

for (Path friendPath : friendsTraverser) {
  counters[friendPath.length()]++;
}
```

Neo4j Spring Data: Object Mapping

```
@NodeEntity
public class UserEntity {

  @GraphId
  private Long id;

  @Indexed(indexName = "user_index", indexType = IndexType.FULLTEXT)
  private String name;

  @RelatedTo(direction = Direction.OUTGOING,
             elementClass = UserEntity.class, type = "follower")
  private Set<UserEntity> follower;


  // ...
}
```

Neo4j Spring Data: Repositories

```
public interface UserRepository extends GraphRepository<UserEntity> {

  @Query(value = "start user=node:user_index(name={0})
                  match user-[:follower]->followingUser
                  return followingUser")
  Set<UserEntity> findFollower(String name);

  // ...

}
```

adesso business. people. technology.

Vielen Dank für Ihre Aufmerksamkeit!

Kai.Spichale@adesso.de

twitter.com/kspichale

spichale.blogspot.de