Entwicklercamp 2012

Track 4 Session 6

NoSQL databases – an overview

Karsten Lehmann | CEO | Mindoo GmbH

About us

- Mindoo is a IBM Business Partner and a Notes/Domino Design Partner
- We focus on the "new" aspects of IBM Lotus Notes development
  - Eclipse/Expeditor plug-ins and rich client applications
  - XPages applications
- As well as web application development for IBM Websphere and Oracle Glassfish
- Karsten Lehmann and Tammo Riedinger
  - Founders of Mindoo GmbH
  - Since 2004, developers of the MindPlan® application for Haus Weilgut GmbH, Mind mapping and project management for Lotus Notes, IBM Award Winner 2008
- More information at:
- http://www.mindoo.com

[mindu:]
Mindoo GmbH

Agenda

- Introduction
- What is NoSQL?
- Classes of NoSQL databases
- Kundera library
- Stocktaking: Where does Lotus Notes stand?
- Summary
- Q&A

When your only tool is a hammer, every problem looks like a nail.

Motivation

- Lotus Notes is not the best solution for every problem
- Lotus Notes is one of the oldest representatives of NoSQL
- For years, more and more dynamic in the NoSQL database field but unfortunately little advancement of NSF
- Through the knowledge of what "the others" can do, one learns to appreciate the Lotus Notes advantages, make justified demands for IBM and has more tools in the repertoire
- The lecture provides a rough NoSQL market overview from the developer's point of view:
  - What are the others capable of?
  - Which problems do I solve with which database?
  - Code snippets to break down inhibitions – everything is so easy :-)

Agenda

- Introduction
- What is NoSQL?
- Classes of NoSQL databases
- Kundera library
- Stocktaking: Where does Lotus Notes stand?
- Summary
- Q&A

What is NoSQL?

- "Not only SQL", not "No SQL"!
- Stands back from the relational scheme in favor of scalability
  → Avoidance of JOINs
- Enables /simplifies scalability and sharding: distribution of data to several linked machines (shards)
- Flexible (natural) storage model
- Large number of more or less active suppliers:
  More than 130 NoSQL databases at http://nosql-database.org !

CAP theorem

- According to Eric Brewer, 2000; since 2002, proven and theorem
- Suitable for the evaluation of database reliability
- Important characteristics with distributed data storage:
  - **Consistency**
    all clients have the same data view at all times
  - **Availability**
    clients can read and write at all times
  - **Partition tolerance**
    the system is working completely despite physical network division
- Theorem: only two of the three properties possible with distributed data storage

CAP theorem

Data models:
Relational
Key/Value-oriented
Column-oriented
Document-oriented
Graph-oriented

Availability:
clients can read and write at all times

A

CA
RDBMS
Neo4j

AP
CouchDB
Cassandra
Lotus Notes
Voldemort

Consistency:
all clients have the same data view at all times

C

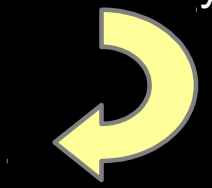
CP
CouchBase
HBase
MongoDb
Redis

P

Partition tolerance:
the system is working completely despite physical network division

CAP theorem

- Leads to abandonment of ACID requirements, moving toward BASE
  - **Atomicity** - implementing a transaction fully or not at all
  - **Consistency** - guaranteeing integrity of the database
  - **Isolation** - transactions are carried out independent of simultaneously running transactions
  - **Durability** - after commitment of the transaction, changes are saved permanently



  - **Basically available** - system is generally available, availability is not guaranteed though
  - **Soft state** - system state can change over time even without data entry
  - **Eventual Consistency** - ultimate consistency: temporary inconsistency of the database is accepted, consistency is restored with a delay only

Agenda

- Introduction
- What is NoSQL?
- Classes of NoSQL databases
  - Key/value-oriented
  - Column-oriented
  - Document-oriented
  - Graph-oriented
- Kundera library
- Stocktaking: Where does Lotus Notes stand?
- Summary
- Q&A

Key/value oriented databases

Key/value oriented

- Representatives
  - Memcached (including YouTube, Wikipedia, Twitter, Flickr)
  - Redis (VMWare employs developers for work on Redis)
  - Voldemort (code of LinkedIn put under Open Source, including Nokia Committer)
- Data model memcached
  - Unstructured map of key/value pairs
  - Analog to a Java HashMap

```
function get_foo(foo_id)
    foo = memcached_get("foo:" + foo_id)
    if (foo!=null) return foo

    foo = fetch_foo_from_database(foo_id)
    memcached_set("foo:" + foo_id, foo)
    return foo
end
```


Key/value oriented

- Redis expands pure key/value store with lists, sets and sorted sets
- Setting/reading values with expiry date (key="resource:lock")

```
SET resource:lock "lockowner1"
//automatic expiration of data
EXPIRE resource:lock 120
GET resource:lock => "lockowner1"
//check time-to-live
TTL resource:lock => 113
```

- Working with lists

```
RPUSH friends "Tom" => ["Tom"]
RPUSH friends "Bob" => ["Tom","Bob"]
LPUSH friends "Sam" => ["Sam", "Tom", "Bob"]
LRANGE friends 0 1 => ["Sam","Tom"]
```


Key/value oriented

- Working with sets (unsorted collections of values)

```
SADD key1 "a"
SADD key1 "b"
SADD key1 "c"
SADD key2 "c"
SADD key2 "d"
SADD key2 "e"
SINTER key1 key2 => "c"
```

- Working with sorted sets (sorted collections of values)

```
ZADD hackers 1940 "Alan Kay"
ZADD hackers 1953 "Richard Stallman"
ZADD hackers 1965 "Yukihiro Matsumoto"
ZADD hackers 1916 "Claude Shannon"
ZADD hackers 1969 "Linus Torvalds"
ZADD hackers 1912 "Alan Turing"
ZRANGE hackers 2 4 => ["Alan Kay","Richard Stallman","Yukihiro
Matsumoto"]
```


Applications

- RAM caches
- Fast saving of logs (e.g. error logs of Load Balancer)
- In-memory MessageQueue for interim buffering of news
- More complex applications with a little creativity, e.g. a Twitter clone:
  → http://redis.io/topics/twitter-clone

Advantages and disadvantages

- Advantages
  - Performance high and predictable
  - Simple data model
  - Clear separation of saving from application logic (because of lacking query language)
- Disadvantages
  - Limited range of functions
  - High development effort for more complex applications

Column-oriented databases

Column-oriented databases

- Representatives
  - Apache Cassandra (formerly Facebook, Twitter for Tweeds, Digg, Netfix)
  - Apache HBase (including Facebook, Twitter)
- Data storage as column values for a row key (row ID)
- Random column values can be stored per row, number limited only by memory space
- Sharding done via row ID; Data of a row have to fit on a server
- Server sorts columns automatically; allows mapping of data lists

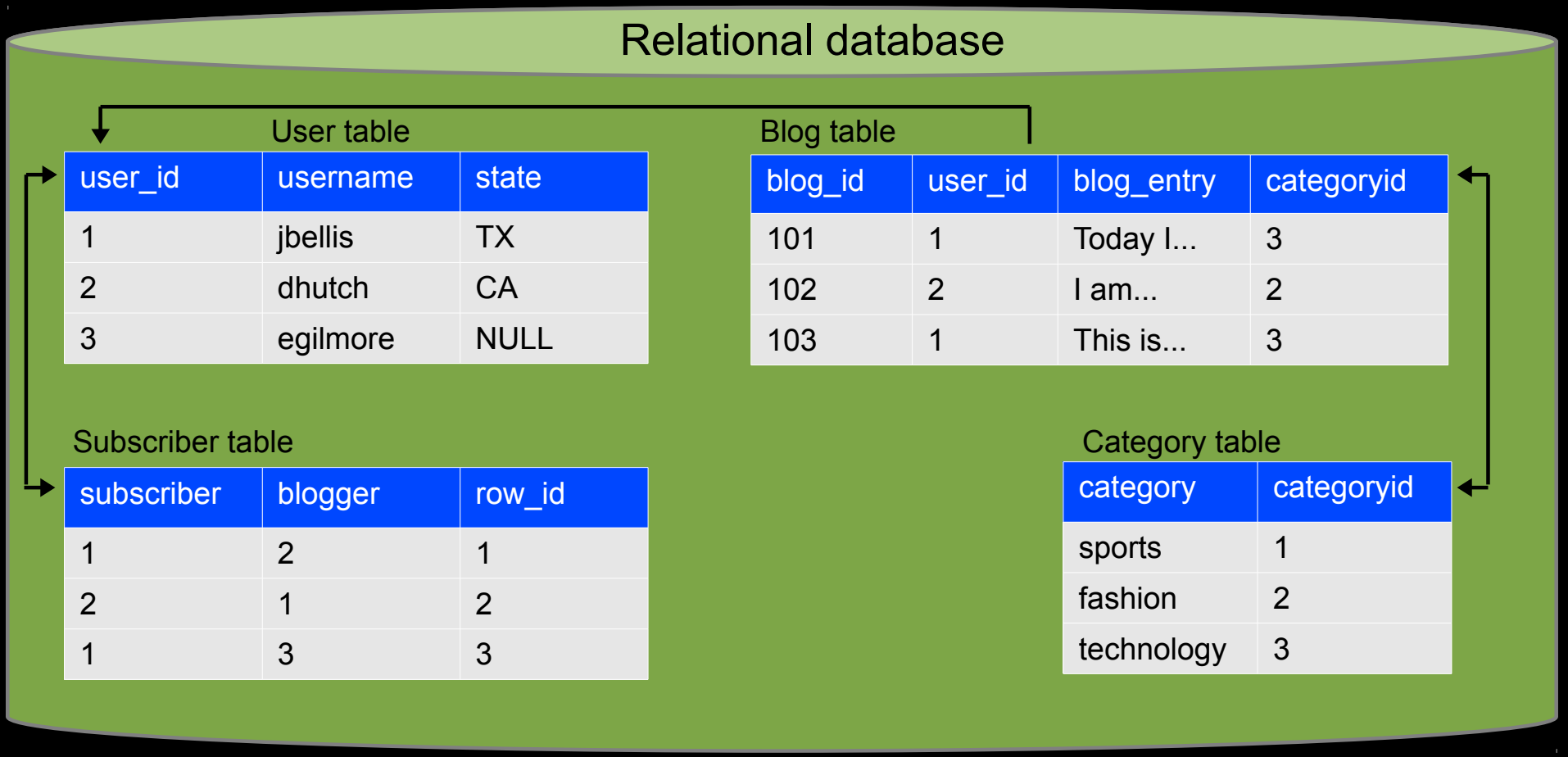
- "If your data model has no rows with over a hundred columns, you're either doing something wrong or you shouldn't be using Cassandra"

Apache Cassandra in detail

- Implemented in Java
- Language support: including Java, JavaScript, Python, PHP, Ruby
- Originally developed at Facebook
- Released since 2008, top level project at Apache since 2010
- Large community of developers and users

Apache Cassandra in detail

- Relational model of a blog



Apache Cassandra im Detail

- Blog implementation with Cassandra

Keyspace „Blog“

Static Column Family (CF) „Users“

jbellis	name	state
	Jonathan	TX
dhutch	name	state
	daria	CA
egilmore	name	state
	eric	

Static Column Family „Blog entries“

92dbeb5	body	user	category
	Today I...	jbellis	tech
d418a66	body	user	category
	I am...	dhutch	fashion
6a0b483	body	user	category
	This is...	egilmore	sports

Dynamic CF „time_ordered_blogs_by_user“

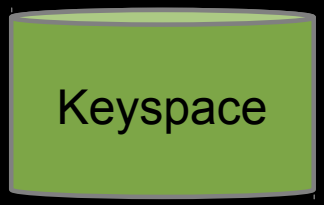
jbellis	1289847840615	
	92dbeb5	
dhutch	1289847840615	
	d418a66	
egilmore		1289847844275
		6a0b483

Dynamic CF „subscribers_of“

jbellis	dhutch	egilmore
dhutch	egilmore	dhutch
egilmore	jbellis	

Apache Cassandra im Detail

- Keyspace
  - Container for several column families
  - E.g. for encapsulation of all data of an application
- Column family
  - Comparable to database table in RDBMS
  - Grouping of several columns
  - Optional determination of column data types
  - Every row can have different column values as oppose to RDBMS
- Static column family
  - Fixed column values, e.g. for properties of an object
- Dynamic column family
  - Random columns, sorted on column name (name e.g. also timestamp or UUID)
  - Query of column areas possible (years 2011-2012)
  - Depending on the case, column name may be sufficient as information, value stays empty

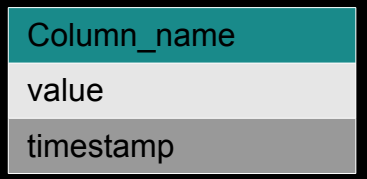


jbellis	name	state
	Jonathan	TX
dhutch	name	state
	Daria	CA
egilmore	name	state
	Eric	

jbellis	dhutch	egilmore
dhutch	egilmore	dhutch
egilmore	jbellis	

Apache Cassandra im Detail

- Column
  - Smallest data unit
  - Tuple with (name, value, time-stamp)
  - Value with highest time-stamp wins
- Special columns
  - Expiring columns: columns with automatic expiration
  - Counter columns: column without timestamp comparison during writing; DB guarantees that value is increased
- Super columns
  - Grouping of several columns in a joint lookup value (e.g. "Address" with columns "Street", "ZIP", "City")
  - Sorting on super column and contained sub columns definable
  - Limited usability because reading sub columns requires loading all sub columns of the super column



Applications

- HBase
  - Data warehousing, analysis of large data stocks with Map/Reduce (Hadoop project)
  - Advantages over Cassandra during reading
- Cassandra
  - Logging of large amounts of data (e.g. price data, sensor data or log files)
  - Advantages over HBase during writing
  - Data written without prior reading process (append only)
  - Data on disk not changed subsequently, compaction cleans outdated data stock
- Generally applications with low query dynamics and very large amounts of data

Advantages and disadvantages

- Advantages
  - Designed for performance
  - Native support for persistent views toward key/value store
  - Sharding: Distribution of data to several servers through hashing row ID
  - Column-oriented systems more efficient than row-oriented during aggregation of a few columns from many rows
- Disadvantages
  - Limited query/filtering options for data
    → Can be compensated through combination with Apache Lucene/Solr: Solandra project
  - High maintenance effort during changing/deleting of existing data because of updating of all lists
  - Less efficient than row-oriented systems during access to many columns of a row

Document-oriented databases

Document-oriented databases

- Representatives
  - MongoDB (including SAP, MTV, Sourceforge, Foursquare)
  - Apache CouchDb (including BBC)
  - Couchbase (including Adobe, BMW, Cisco, Zynga)
  - Lotus Notes
- Storage of data in the form of documents, frequently mapped in JSON format

```
{
 title : 'This is a blog post',
 author : 'Peter',
 content : 'It's working!'
}
```

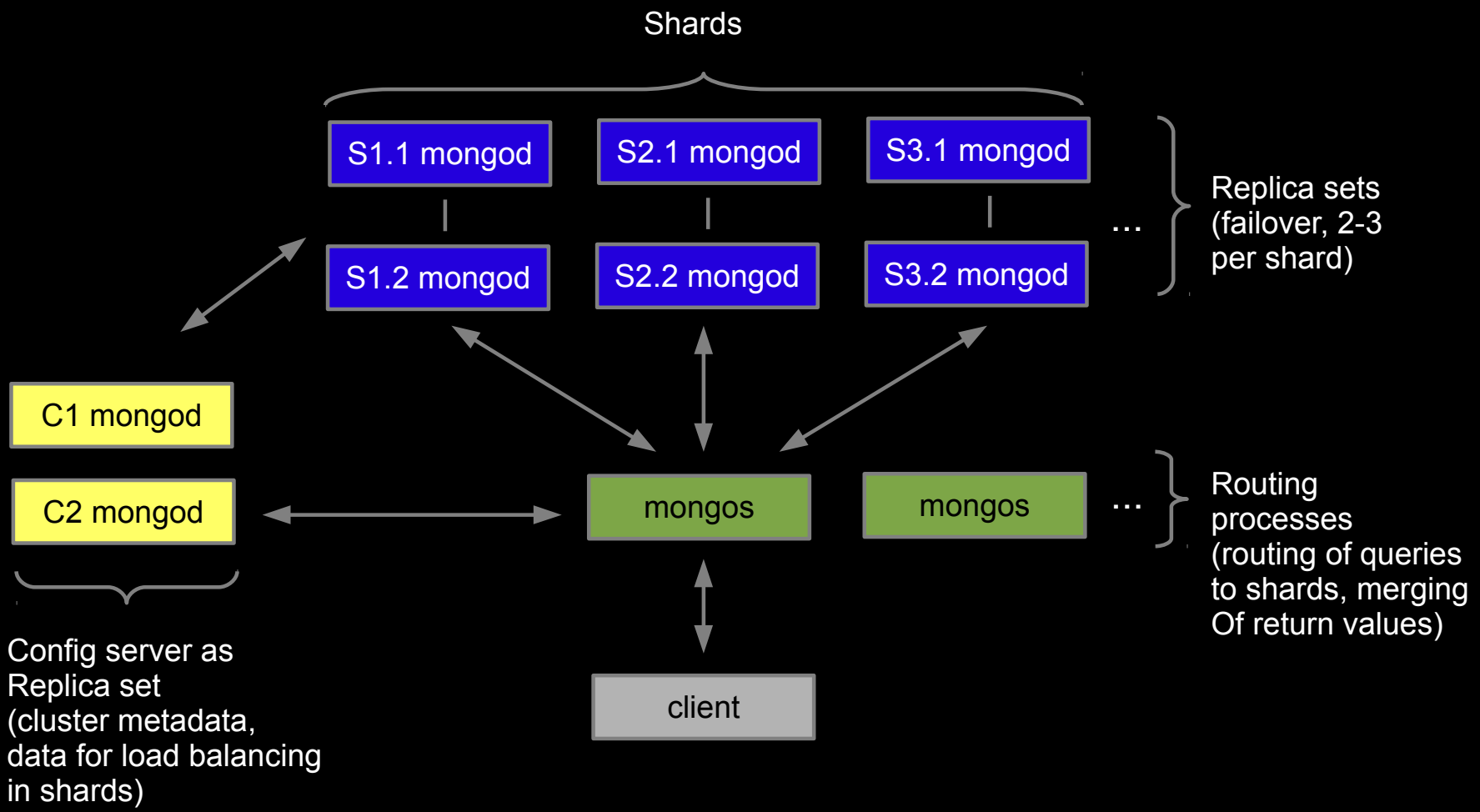
- Documents grouped as collections/views
- Queries ad hoc or via persistent view index

MongoDB in detail

- Implemented in C++
- Language support: including Java, JavaScript, Python, PHP, Ruby
- Saved as binary JSON: BSON
- Supports sharding
  - done via configurable document fields
  - System automatically balances data between shards
  - Shards can be amended during runtime
  - Selection of sharding keys essential for performance!
  - Avoidance of hotspots, i.e. servers with high I/O load
  - Simultaneous index updating after changes on as few machines as possible
    → Finding a compromise

MongoDB in detail

- Sharding architecture:



MongoDB in detail

- Create connection to DB

```
Mongo connection = new Mongo(server,port);
//collection gets created automatically
DBCollection collection =
connection.getCollection(“posts”);
```

- Collect data (JavaScript syntax)

```
collection.insert({
 title : 'This is a blog post',
 author : 'Karsten',
 ts: 1331915835960,
 content : 'It's working!'});
```


MongoDB in detail

- Collect data (Java syntax)

```
DBObject post = BasicDBObjectBuilder.start()
 .add("title", "This is a blog post")
 .add("author", "Karsten")
 .add("ts", 1331915835960)
 .add("content", "It's working!"))
 .get();

collection.insert(post);

System.out.println(post.get("_id"));
=> 4a9700dba5f9107c5cbc9a9c
```

- Or different constructors of BasicDBObject

MongoDB in detail

Update data

```
//$ commands for advanced query/update features
collection.update({_id : post._id},
{
 mod_ts : 1331915836050
 $push : {comments :
  {
      author : 'Tammo',
      comment : 'Cool!'
  }
}});
```

- First argument: query for document
  Second argument: new field values
- $push adds array values or creates array

MongoDB in detail

- Data readout
- Comprehensive query language for data selection
  - $gt, $gte, $lt, $lte, $eq, $neq, $exists, $set, $mod, $where, $in, $nin, $inc, $push, $pull, $pop, $pushAll, $popAll

```
collection.find({ x : {$gt : 4}}) //x greather than 4
```

- Index creation for document values speeds up access to / search in collections

```
//create index for more performance:
collection.ensureIndex({"comments.author": 1})

commentsByPeter = collection.find({
        "comments.author" : "Peter"
});
```


MongoDB in detail

- Reading data with paging

```
collection.ensureIndex({ "author" : 1,
 "ts" : -1,
 "comments.author" : 1 });

var cursor=collection.find({author:'Peter'})
  .sort({ts:-1})
  .skip(pageNum * resultsPerPage)
  .limit(resultsPerPage);

while (cursor.hasNext() ) {
 printArticle( cursor.next() );
}
```


Demo
Using MongoDB in an XPages application

Advantages

- Intuitive data structure, already familiar from Lotus Notes
- Simple "natural" modeling of requests with flexible query functions
  - Views per Map Reduce in Couch
  - Mongo Query language
- MongoDB/CouchBase: sharding for Big Data

Advantages

- MongoDB
  - Query language similarly dynamic as SQL
  - GridFS as add-on to MongoDB to save large binary files incl. sharding/replication in a cluster (max. document size otherwise 4 MB)
  - Geo-spatial indexing built-in (search of next gas station at [longitude,latitude])
- CouchDB
  - Offline synchronization similar to Lotus Notes
- CouchBase Mobile
  - iOS/android port of the database, communicates with CouchBase server in the Cloud

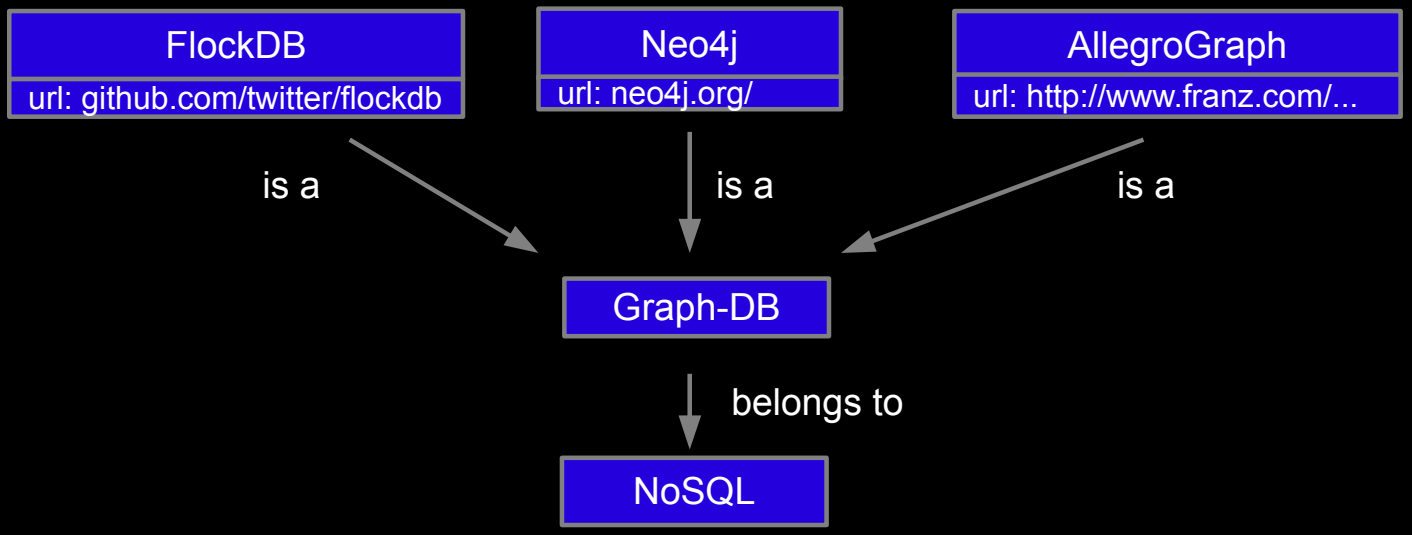
Disadvantages

- CouchDB: future uncertain, developer team focuses on new CouchBase DB
- MongoDB: no solutions for offline data storage on mobile devices
- Compared to column-based database, probably higher hardware demands because of more dynamic DB queries in part without data preparation
- Compared to RDBMS, redundant storage of data (denormalization) in favor of higher performance

Graph-oriented databases

Graph-oriented databases

- Representatives
  - FlockDB (developed by Twitter for Social Graph)
  - Neo4j (including Deutsche Telekom, Adobe, studiVZ, Cisco)
  - AllegroGraph (including Kodak, Pfizer and in various Semantic web projects)
- Databases optimized for highly networked data
- Graph from nodes and relationships between nodes
- Nodes and relationships can have properties



Neo4j in detail

- Implemented in Java
- Unlike previous NoSQL examples, ACID compliant
- Easily embedded in individual applications

```
EmbeddedGraphDatabase graphDb =
            new EmbeddedGraphDatabase( DB_STORAGEDIR );
//
//work with graphDb here
//
graphDb.shutdown();
```


Neo4j in detail

DB access encapsulated in transactions

```
Transaction tx = graphDb.beginTx();
try {
    // create nodes and relationship
    firstNode = graphDb.createNode();
    firstNode.setProperty( "message", "Hello, " );
    secondNode = graphDb.createNode();
    secondNode.setProperty( "message", "World!" );

    relationship = firstNode.createRelationshipTo(
                    secondNode, RelTypes.KNOWS );
    relationship.setProperty( "message", "brave Neo4j " );

    // read data
    System.out.print( firstNode.getProperty( "message" ) );
    System.out.print( relationship.getProperty( "message" ) );
    System.out.print( secondNode.getProperty( "message" ) );

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

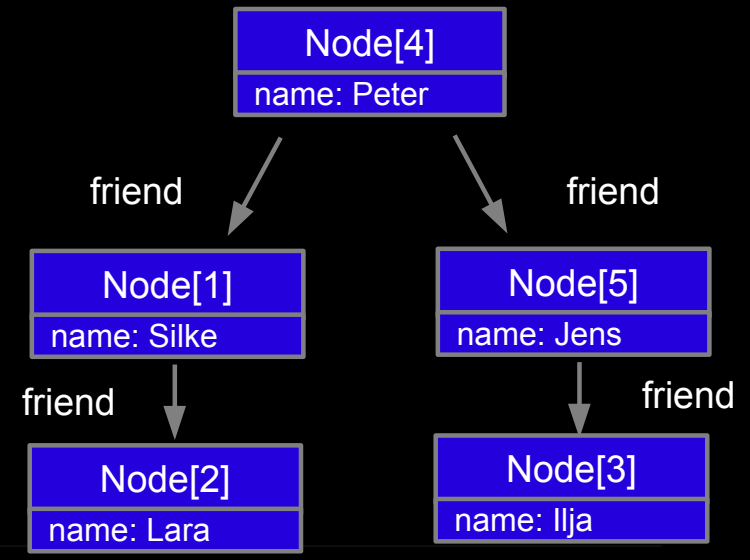

Neo4j in detail

- API functions for traversing graphs and searching for all /the shortest paths between two nodes
- Query language: Gremlin (Groovy based traversing language) and Cypher (declarative graph query language)
- Cypher: Find all friends of Peter's friends who are not his friends

```
START peter=node:node_auto_index(name = 'Peter')
MATCH peter-[:friend]->()-[:friend]->fof
RETURN peter, fof
```

- Result:

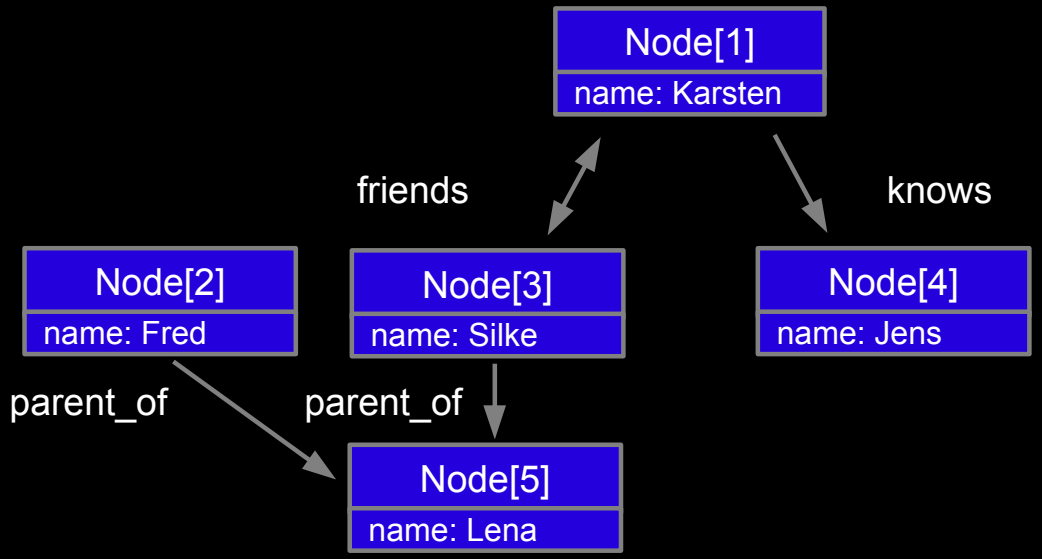
peter	fof
Node[4] {name->"Peter"}	Node[2] {name->"Lara"}
Node[4] {name->"Peter"}	Node[3] {name->"Ilja"}



Neo4j in detail

- Cypher: Deliver all persons that I know and their children if they have any:

```
START me=node(1)
MATCH me-->person-[?:parent_of]->children
RETURN person, children
```



Applications

- Modeling of social networks
  - Friend recommendations
- Examination of access rights
  - Person X - is included in → group A
  - Group A - is included in → group B
  - Group B - is included as editor in → ACL of names.nsf
  - What type of access does the user have to which databases?
- Product recommendations
  - What music that I haven't bought yet do my friends like?
- Route planning

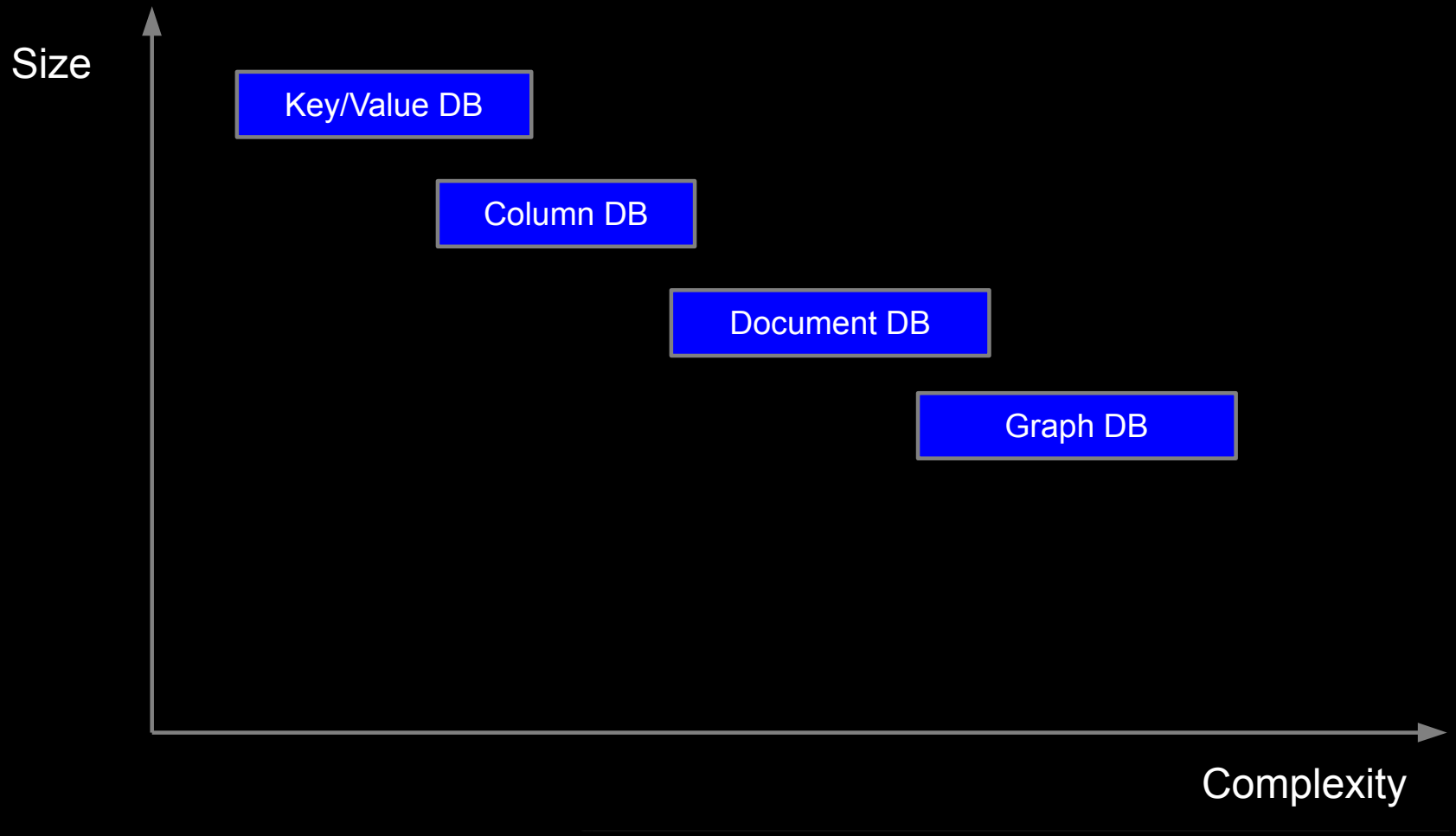
Demo
Neo4j embedded in XPages application

Advantages and disadvantages

- Advantages
  - Very compact modeling of networked data
  - Large performance advantages over RDBMS depending on application case
  - Neo4j
    - ACID compliance
    - Problem-free embedding in other applications
- Disadvantages
  - Neo4j
    - Unlike FlockDB, no sharding built-in but master-slave replication
    - „This can handle billions of entities…but not 100B“
    - License fees for OEM use

Overview of NoSQL size/complexity

- Increasing DB complexity lowers the possibility of distributing data efficiently



Agenda

- Introduction
- What is NoSQL?
- Classes of NoSQL databases
- Kundera library
- Stocktaking: Where does Lotus Notes stand?
- Summary
- Q&A

Kundera – the common denominator

- JPA* compatible tool kit for saving data in RDBMS, HBase, Cassandra and MongoDB
- Simple persistence of data with the help of annotated POJOs**
- Maps JPA query language on DB functions
  - Uses Apache Lucene as external indexer for missing functions
- Change of NoSQL database merely a configuration change

*Java Persistence API
**Plain Old Java Objects

Kundera – the common denominator

```
@Entity
@Table(name = "users", schema = "KunderaExamples@cassandra_pu")

public class User {
    @Id
    private String userId;
    @Column(name="fullname")
    private String fullName;

    public User() {

    }
    public String getUserId() {
        return userId;
    }
    public void setUserId(String userId) {
        this.userId = userId;
    }
    public String getFullName() {
        return fullName;
    }
    public void setFullName(String fullName) {
        this.fullName = fullName;
    }
}
```


Kundera – the common denominator

```
User user = new User();
user.setUserId("0001");
user.setFullName("Hans Müller");

EntityManagerFactory emf =
       Persistence.createEntityManagerFactory("cassandra_pu");
EntityManager em = emf.createEntityManager();


//save object
em.persist(user);

//load object
User anotherUser = em.find(User.class, "0002");


em.close();
emf.close();
```


Kundera – Advantages and disadvantages

- Advantage
  - Investment-safe development because of independence of database
  - Comprehensive queries even with column-oriented DB, Lucene index storable in Cassandra itself
  - Simplified data transfer between NoSQL databases
- Disadvantage
  - Less influence on definite data storage; poss. lowest common denominator of the features
  - Kundera Open Source, but the community is rather small

Agenda

- Introduction
- What is NoSQL?
- Classes of NoSQL databases
- Kundera library
- Stocktaking: Where does Lotus Notes stand?
- Summary
- Q&A

Features of the others that Lotus Notes is missing?

- Storage of more than 16/32/64 KB data in fields
- Modern Java APIs
  - designed for performance
  - no recycling
  - Java generics
  - attachment streaming
  - access to transactions in the code
- Ad hoc database queries
  - UnQL, Mongo Query Language, Map Reduce
- Web 2.0 compatible license model for startups (free without support)
- Big Data support: Removal of the 64 GB limits, sharding
- Geo-spatial indexing

Features of Lotus Notes that the others are missing?

- NSF
  - Built-in storage of attachments in documents

- Miscellaneous
  - All-in-one solution of useful individual components
  - User/Group directory
  - HTTP server
  - Full text engine incl. indexing of file formats
  - Mail server / calendaring and scheduling
  - Application server
  - Read/write access rights on document level
  - Encryption

Agenda

- Introduction
- What is NoSQL?
- Classes of NoSQL databases
- Kundera library
- Stocktaking: Where does Lotus Notes stand?
- Summary
- Q&A

Summary

- A lot of dynamics in the NoSQL market
  - no product is perfect
- Comparison of NSF with other databases disillusioning
  - but Lotus Notes is far more than just a database
- Because of the number of system components, it is more a Swiss knife than a hammer
- Selection of DB platform no either-or
  - finding the right tool for an application case
  - combinations of DBs also possible!
- Not forgetting SQL:
  Who really wants eventual consistency
  for financial transactions? :-)

- Introduction
- What is NoSQL?
- Classes of NoSQL databases
- Kundera library
- Stocktaking: Where does Lotus Notes stand?
- Summary
- Q&A

Thanks!

Time for questions