swiftmind

NoSQL in Enterprise Java Applications

Patrick Baumgartner

NoSQL Search Roadshow | Munich | 16.04.2013

Agenda

- Speaker Profile
- New Demands on Data Access
- New Types of Data Stores
- Polyglot Persistence
- Integrating NoSQL Data Stores
- Spring Data Overview
- Example with MongoDB
- Example with Neo4j
- Highlights and Challenges
- Q & A

Speaker Profile

Patrick Baumgartner

- Senior Software Consultant | Partner @ Swiftmind

- Spring Framework, OSGi, Neo4j
- Spring Trainer
- Co-Autor von „OSGi für Praktiker“

- Agile Methodologies & Practices
- Scrum.org Trainer - PSD Java

- @patbaumgartner

Swiftmind

Your experts for Enterprise Java

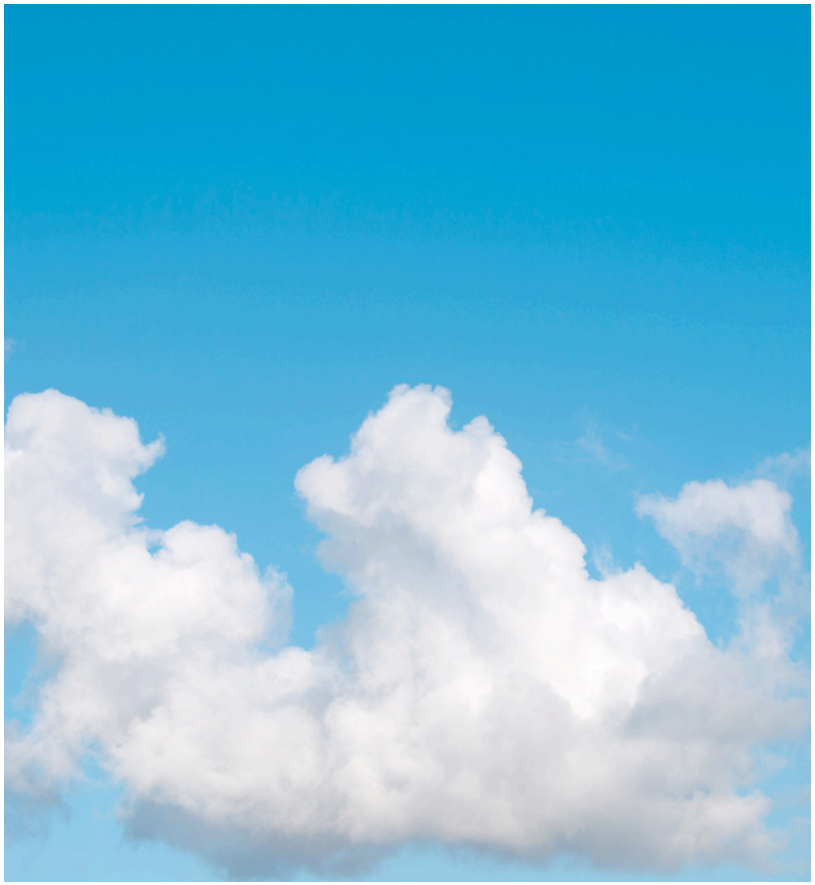
Areas of expertise

- Java EE
- Spring Framework
- OSGi
- Agile Methodologies
- Software Engineering Best Practices

Headquarter

- Zürich, Schweiz
- @swiftmind
- http://www.swiftmind.com

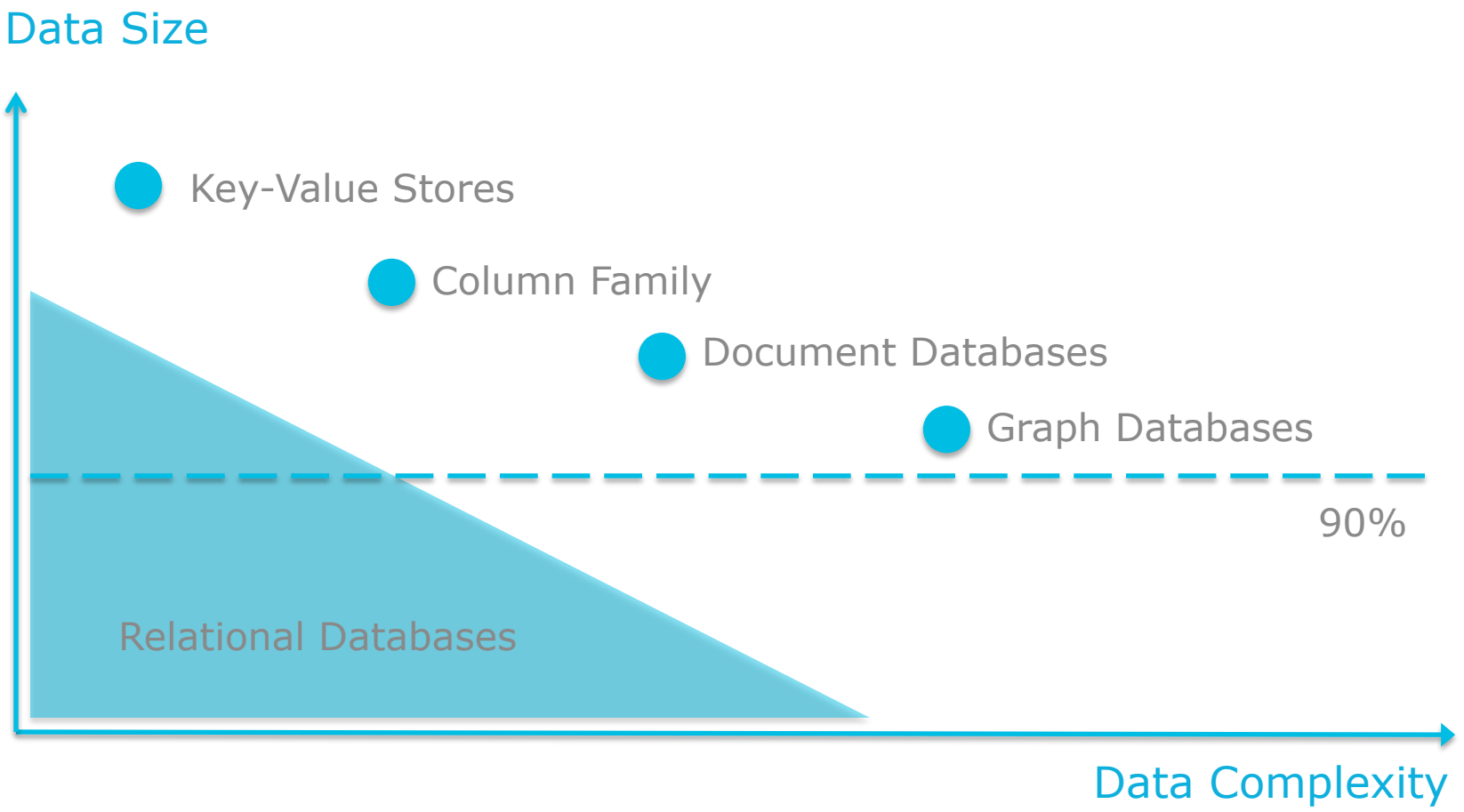
New Demands on Data Access



- Structured and unstructured data
- Massive amounts of data
- Inexpensive horizontal scaling
- Apps and data in the cloud
- Social network features
- …

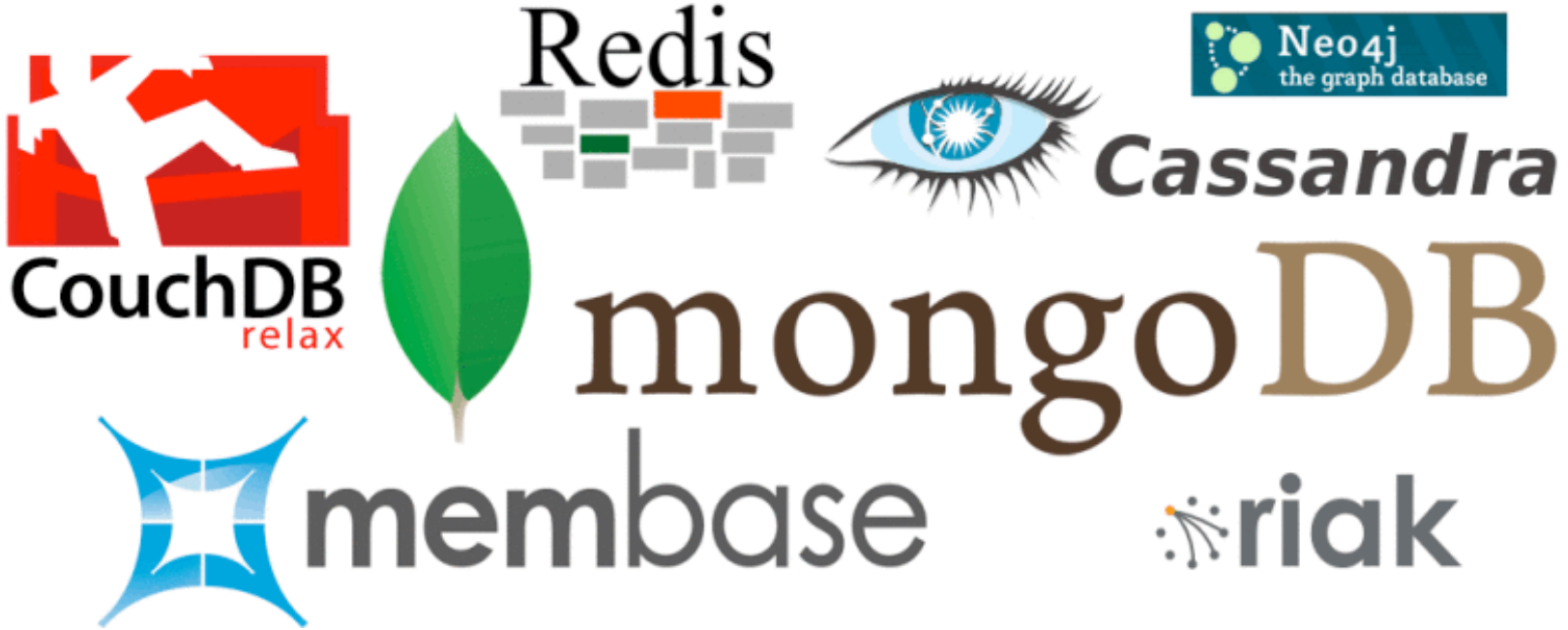
New Types of Data Stores

NoSQL

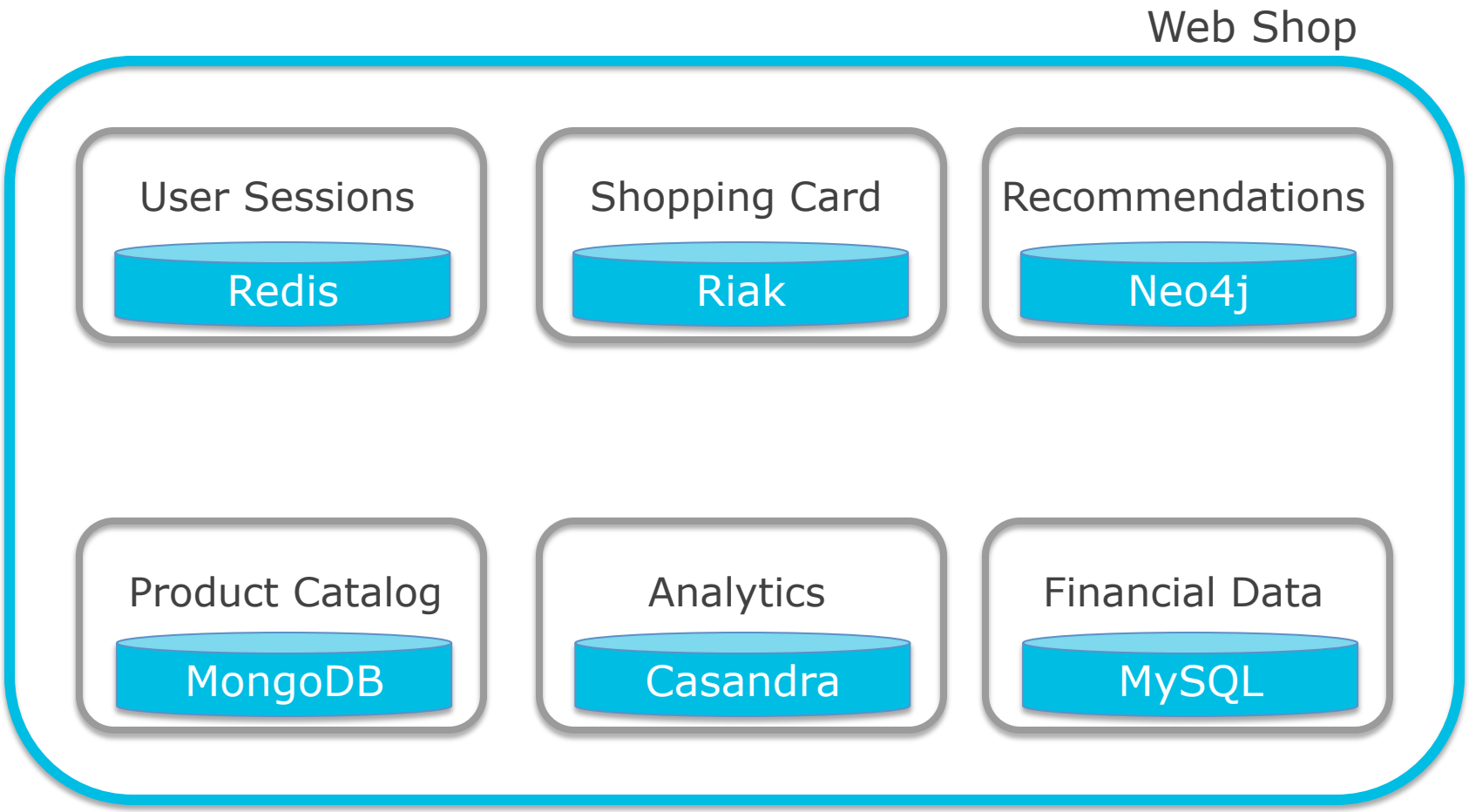

Data Size
Key-Value Stores
Column Family
Document Databases
Graph Databases
90%
Relational Databases
Data Complexity

Polyglot Persistence

One single data store for your application?



Polyglot Persistence



Integrating NoSQL Data Stores #1

We are not architects!

http://www.flickr.com/photos/sakeeb/4087246274

Integrating NoSQL Data Stores #2

Don’t re-invent the wheel!

http://www.flickr.com/photos/dmott9/5921728819

Integrating NoSQL Data Stores #3

Let’s use Spring!

http://www.springsource.org/spring-data

Spring Data

- Same goals as the Spring Framework
  - Productivity improvement
  - Programming model consistency
  - Portability

- Umbrella for subprojects, specific to a given database
- Mapping support for Java domain objects

- http://www.springsource.org/spring-data
- http://github.com/SpringSource/spring-data-XXX

Spring Data – Overview #1

Relational Databases

- JPA
- JDBC Extensions

Big Data

- Apache Hadoop

Data Grid

- GemFire

HTTP

- REST

Key Value Stores

- Redis

Spring Data – Overview #2

Document Stores

- Mongo DB
- Couchbase *

Graph Databases

- Neo4J

Column Stores

- Hbase

Search

- Apache Solr *
- Elastic Search *

Common Infrastructure

- Commons

* Community Project

Spring Data – Key Features

- Low level data access API abstraction
  - MongoTemplate, RiakTemplate, Neo4jTemplate …
  - Exception translation
  - Transaction management
- Object Mapping (Java to data store)
- Generic Repository Support
  - Basic CRUD, dynamic finders, pagination and sorting
- Spring namespaces
- Cross Store Persistence Programming Model
  - @Entity, @Document, @NodeEntity …

Spring Data MongoDB – Example

Documents


RICHARD
Padlocks, Alcove, Park
ALSO DOUBLE
NEW ROAD, Willenhall,
ESTABLISHED 1792.
Messrs. Varley, Brothers.
Manufacturers of
Locks, Alcove, Park Gate & Lighter Lock
& BAR PADLOCKS, & SETS IN EVERY VARIETY
Birmingham, 28/6 1924
900
N, SON & CO.
AGENTS FOR
CONTINENTAL STEAMSHIP CO. LTD.
STEAM SHIP CO.

Spring Data MongoDB – Document

```
@Document
public class Person {
  @Id
  private int id;
  private String name;
  private int age;
  // getters/setters…
}
```

Stored JSON:

```
{ "_id" : ObjectId("4f9886290364b533b3acd4ce"),
  "_class" : "com.example.Person",
  "name" : "Bob",
  "age" : 33
}
```

Spring Data MongoDB – Configuration

```
<bean id="mongoTemplate" class="org.springframework.data.document.mongodb.MongoTemplate">
    <constructor-arg name="mongo" ref="mongo"/>
    <constructor-arg name="databaseName" value="test"/>
    <constructor-arg name="defaultCollectionName" value="HelloMongo"/>
</bean>

<!-- Factory bean that creates the Mongo instance -->
<bean id="mongo" class="org.springframework.data.document.mongodb.MongoFactoryBean">
    <property name="host" value="localhost"/>
</bean>
```

Spring Data MongoDB – Repository

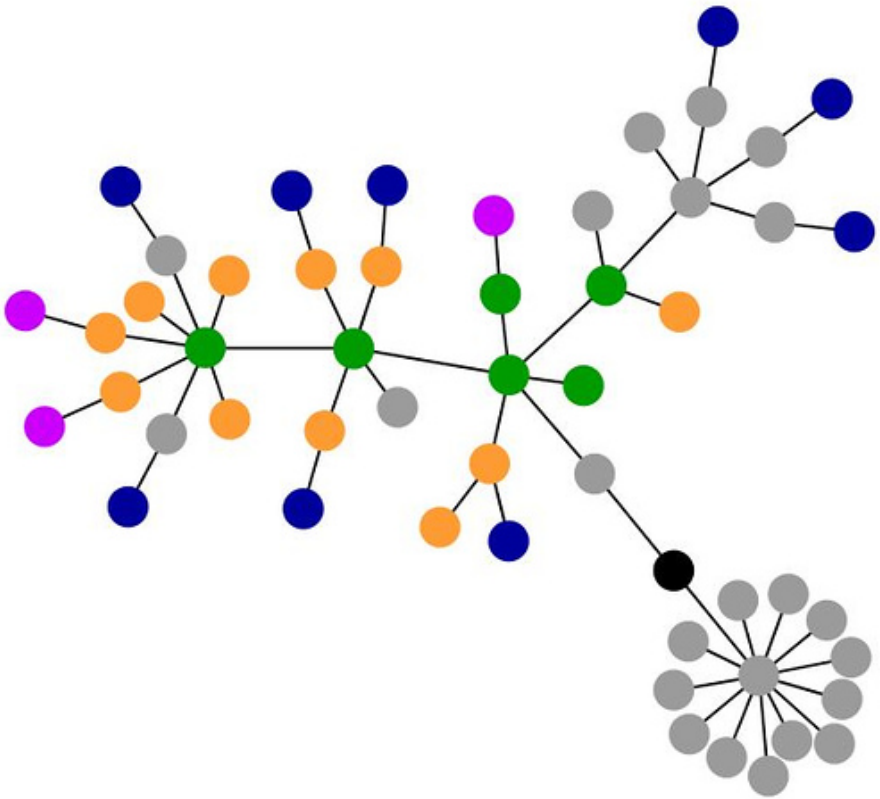
```
@Repository
public class MongoPersonRepository implements BaseRepository<Person> {

  @Autowired
  MongoOperations mongoTemplate;

  Person createPerson(String name, int age){
    if(!mongoTemplate.collectionExists(Person.class)){
      mongoTemplate.createCollection(Person.class);
    }
    Person p = new Person(name, age);
    mongoTemplate.insert(p)
    return p;
  }

  ...
}
```

Spring Data MongoDB – Repository #2

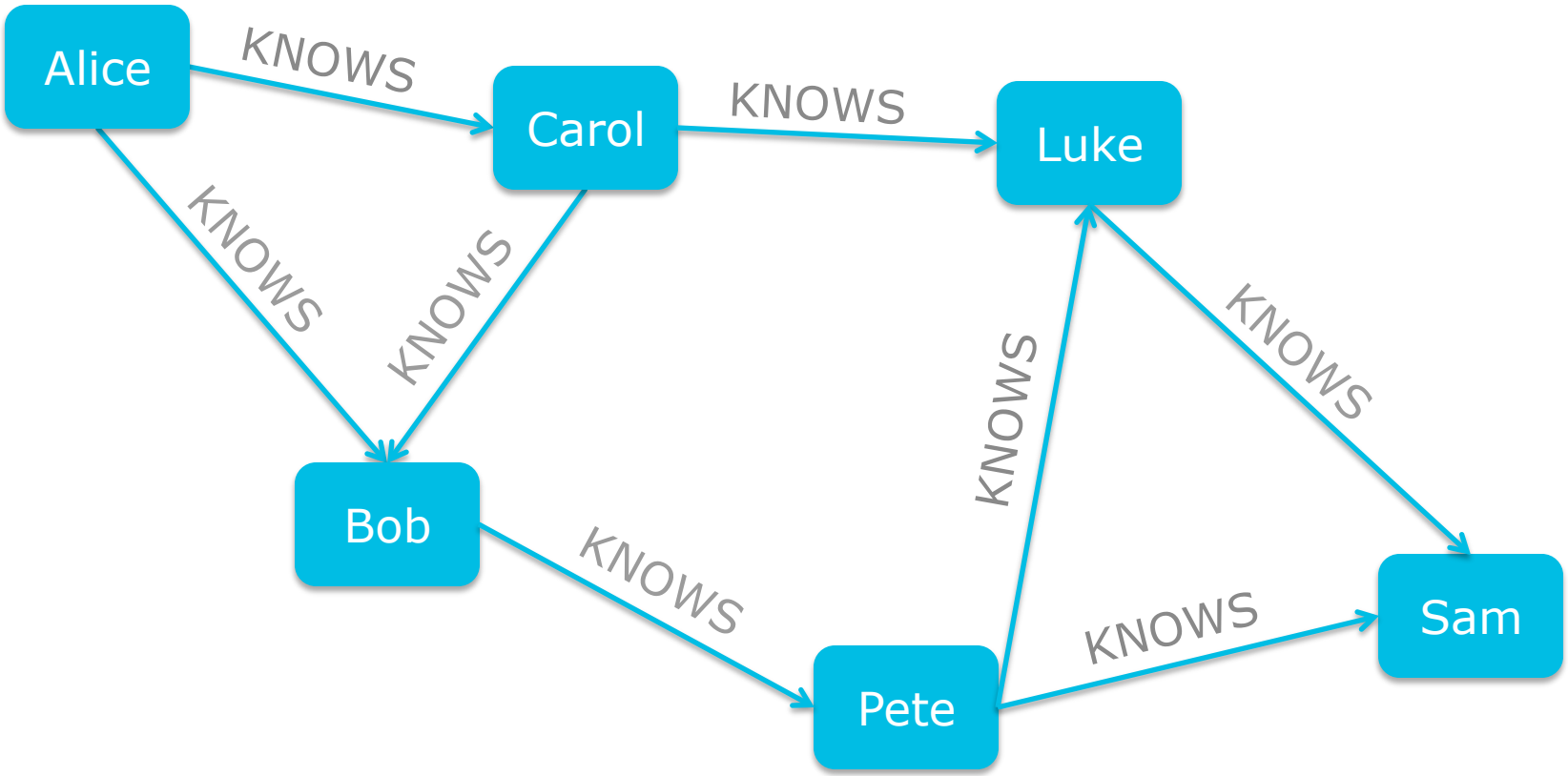
```
@Repository
public interface MongoPersonRepository implements CrudRepository<Person, Long>
{
  Page<Person> findByNameContaining(String name, Pageable p);


  @Query("{ ?0 : ?1 }")
  List<Product> findByAttributes(String key, String value);
}
```

Spring Data Neo4j – Example

Graph

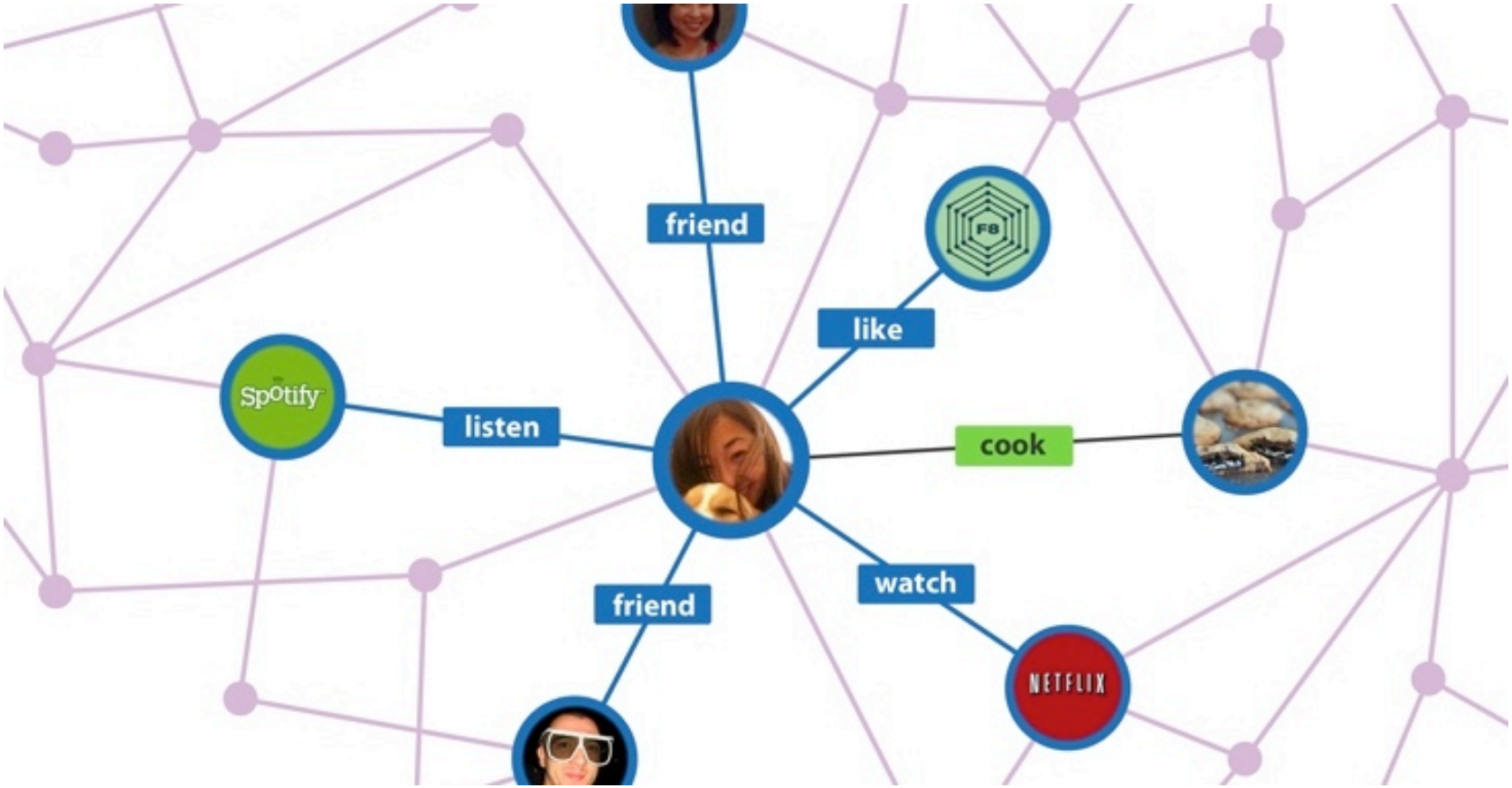


Social Data

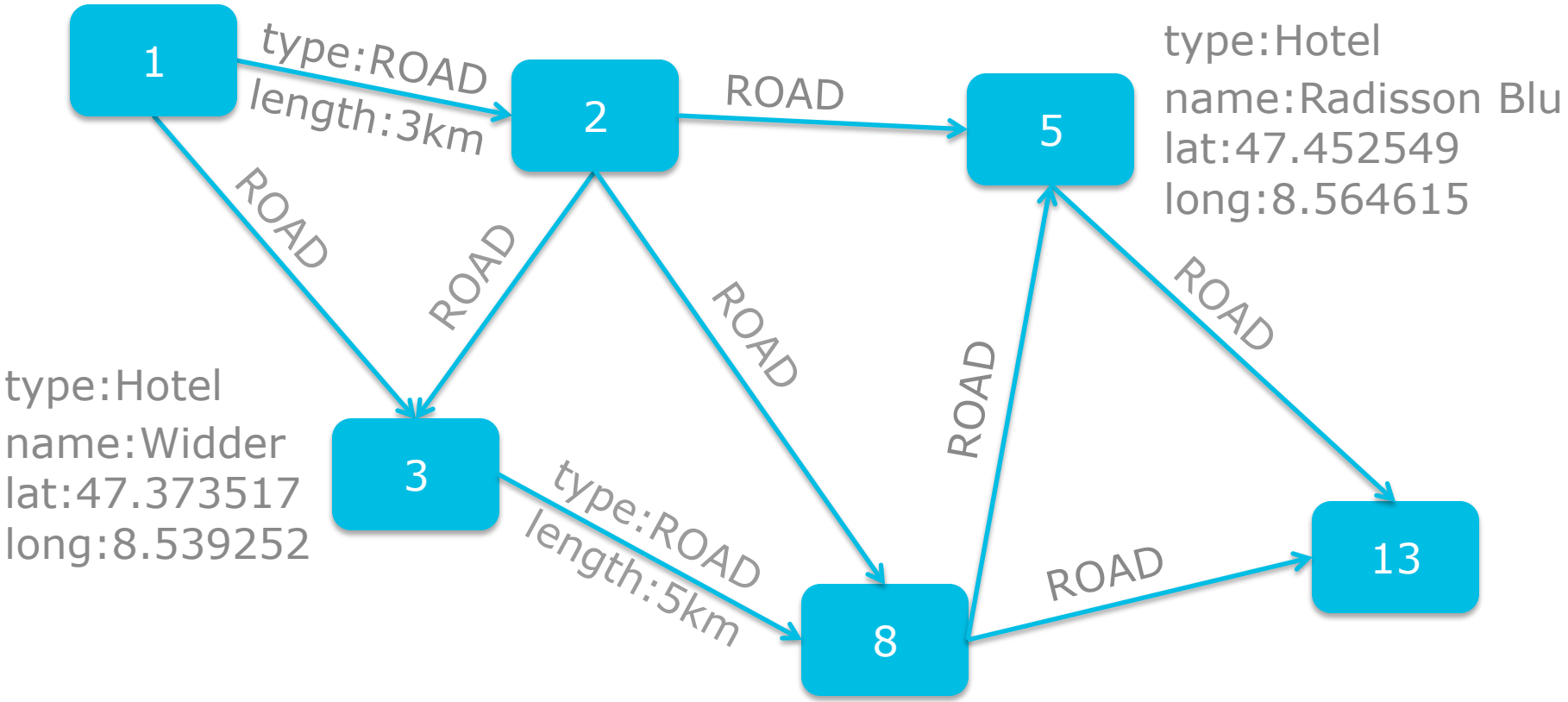

Alice
KNOWS
Carol
KNOWS
Luke
KNOWS
KNOWS
KNOWS
KNOWS
KNOWS
Bob
KNOWS
Pete
KNOWS
Sam

Social Graph

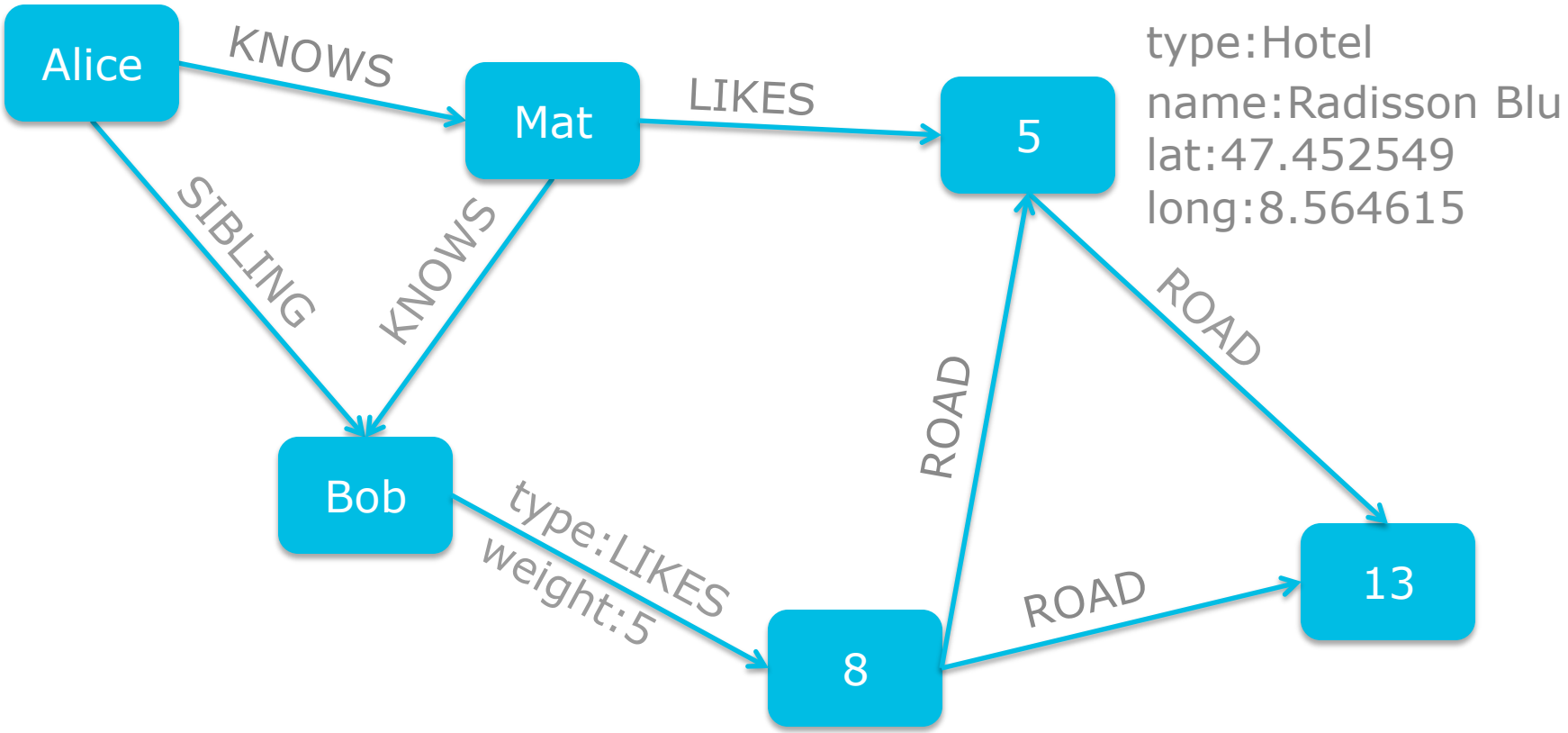
The only use case?



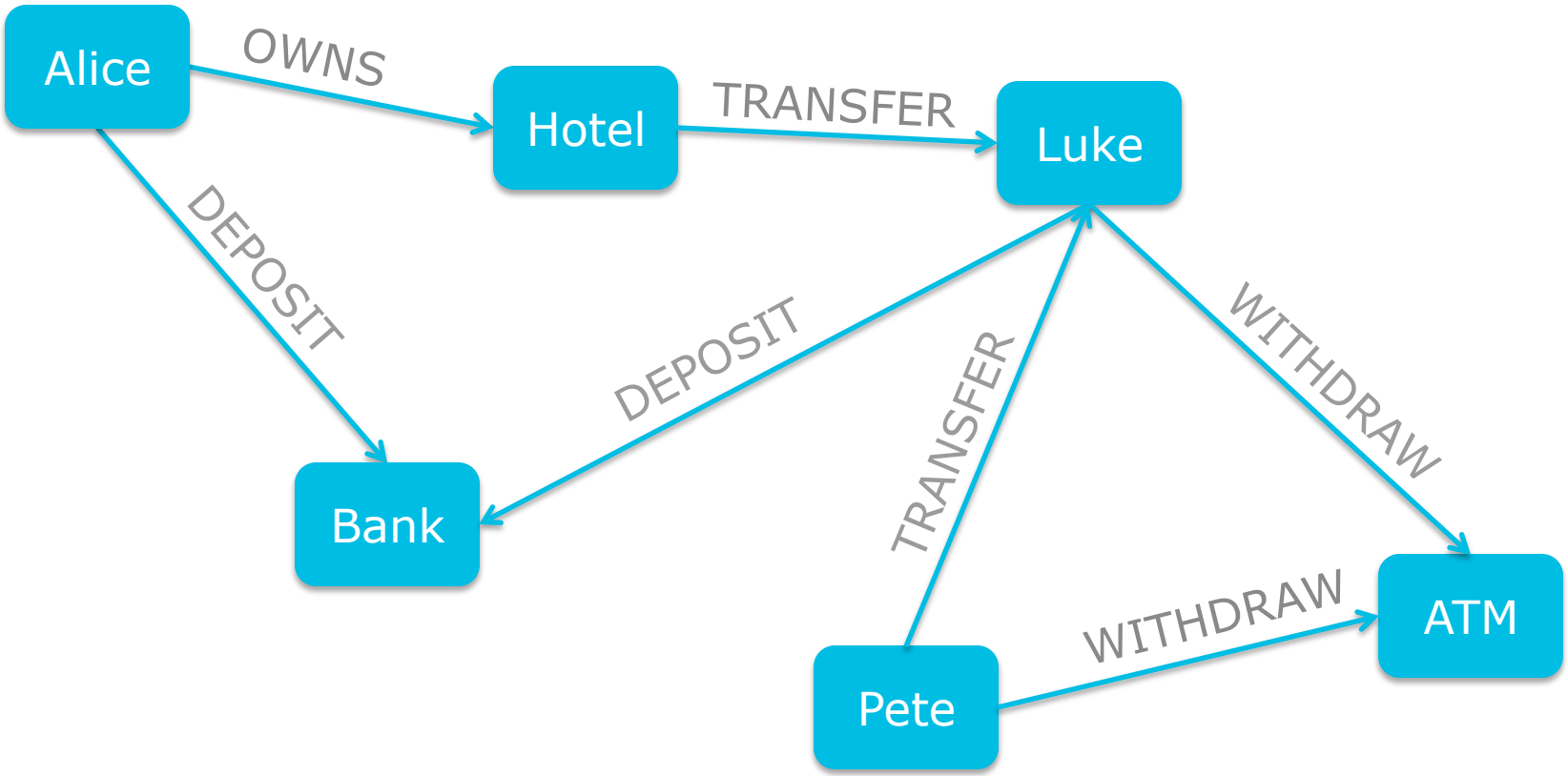
Spatial Data



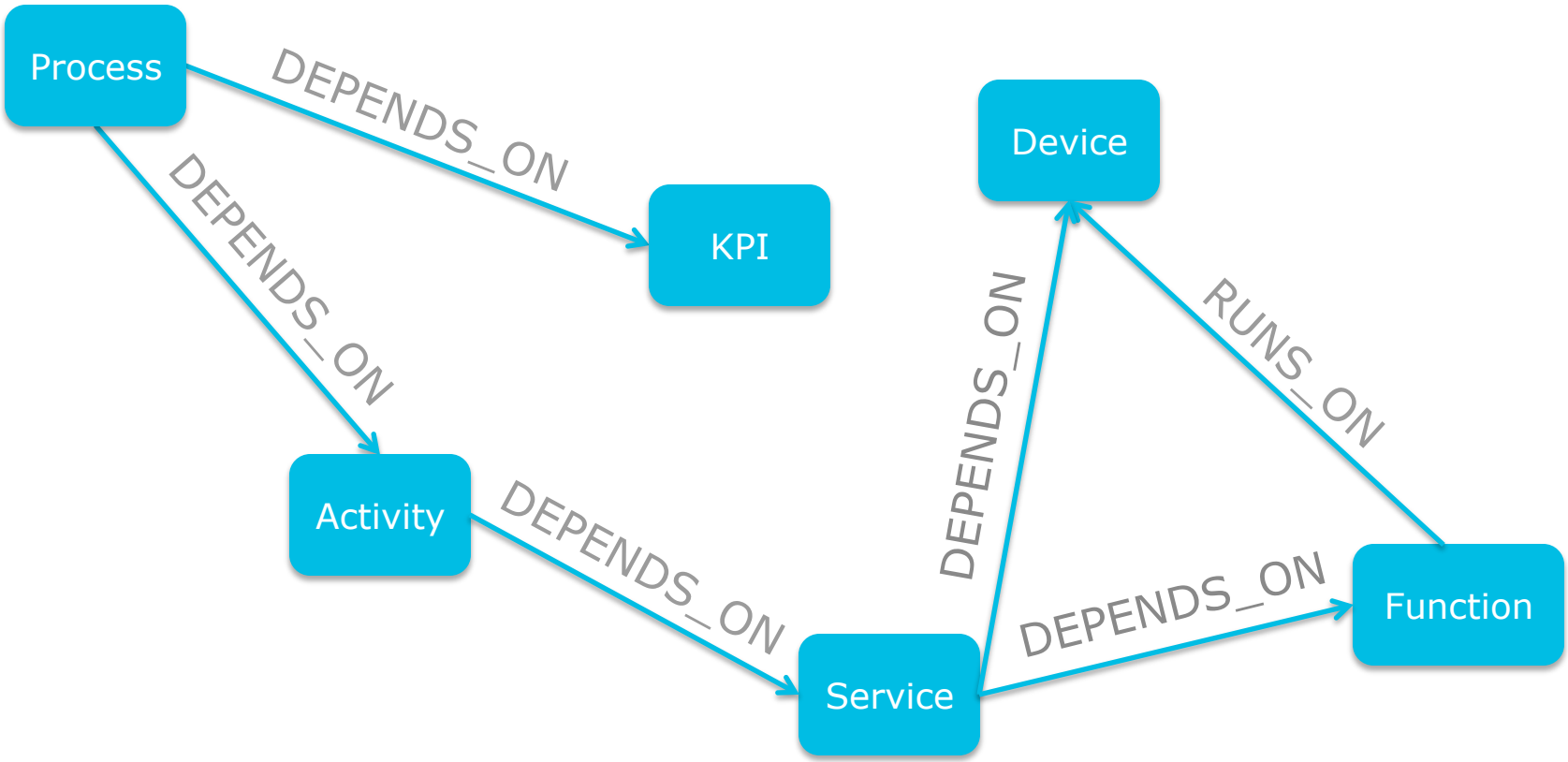
Social & Spatial Data



Financial Data



Your business domain

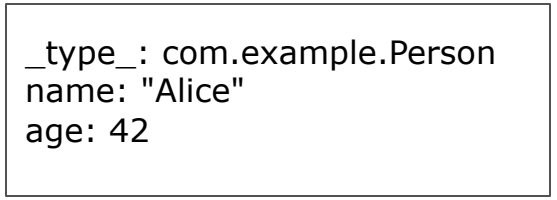


Spring Data Neo4j (SDN)

- POJOs mapped as nodes or relationships – type safe
- Works directly Database, typically embedded mode
- Data is fetched very fast and lazy
- Stores everything within a @Transaction
- Uses heavily AspectJ magic to enhance the POJOs
- …

Spring Data Neo4j – NodeEntity

```
@NodeEntity
public class Person {
  private String name;
  private int age;
  // getters/setters…
}

Person alice = new Person();
alice.setName("Alice");
alice.setAge(42);
alice.persist();
```

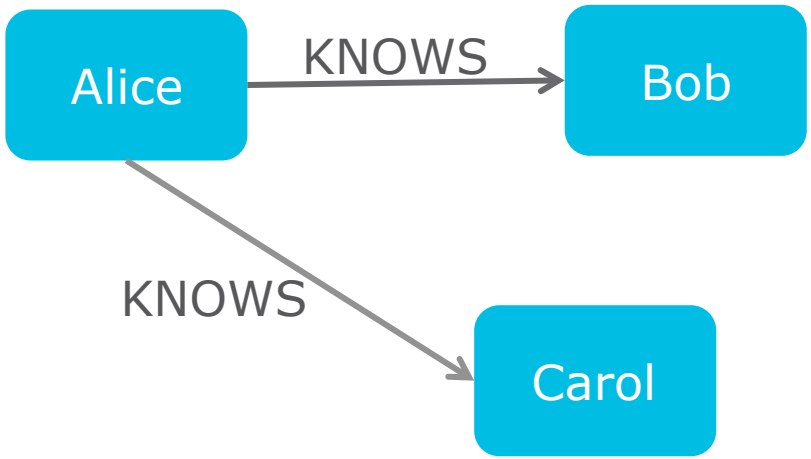


Spring Data Neo4j – Relationship

```
@RelationshipEntity
public class Knows {
  private int sinceYear;
  public Knows since(int year) {
    this.sinceYear = year;
    return this;
  }
}

@NodeEntity
public class Person {
  public Known knows(Person p) {
    return this.relateTo(p, Knows.class, "KNOWS");
  }
}
Person alice = ...;
Person bob ...;
alice.knows(bob).since(2012);
```

Spring Data Neo4j – NodeEntity

```
@NodeEntity
public class Person {
  private String name;
  private int yearOfBirth;
  @RelatedTo(type = "KNOWS", direction = Direction.OUTGOING)
  private Set<Person> knownPersons;

  public void knows(Person p) {
    knownPersons.add(p);
  }
  public Set<Person> getFriends() {
    return knownPersons;
  }
}
Person alice = ...;
alice.knows(bob);
alice.knows(carol);
```



Spring Data Neo4j – Repository

```
public interface PersonRepository extends
GraphRepository<Person> {
  @Query("start person = {0} match ... return ...")
  Iterable<Product> getOwnedServices(Person person);
  Iterable<Person> findByName(String name);
  Iterable<Person> findByNameLike(String name);
}

@Autowired
PersonRepository personRepository;
Person alice = personRepository.findByName("Alice");
```

Spring Data Neo4j – Querying

- Several possibilities implemented to query the graph
  - Neo4j API
  - Traversal descriptions
  - Graph algorithms
  - Index queries
  - Cypher queries

Highlights and Challenges

Highlights

- Vibrating community, great support on mailing lists
- Using the right tool for the right job
- New technologies differentiate our product/service from competitors
- Sources on github.com – Pull requests for improvements
- Very good overall performance
- We can use Spring!
- Abstraction layer makes “driver” changes easy

Challenges

- Developer Onboarding
- Schema changes -> migrations
- Some topics not solved yet -> e.g. versioning
- Not many show cases documented in detail
- Sizing for scaling based on real world measurements
- Massive amount of data
- Several framework and data store releases per year
- Bugs, but no show stoppers
- Keeping documentation up-to-date, data model changes

Spring Data

Information from the source

Hive Pig JDBC HBase JPA

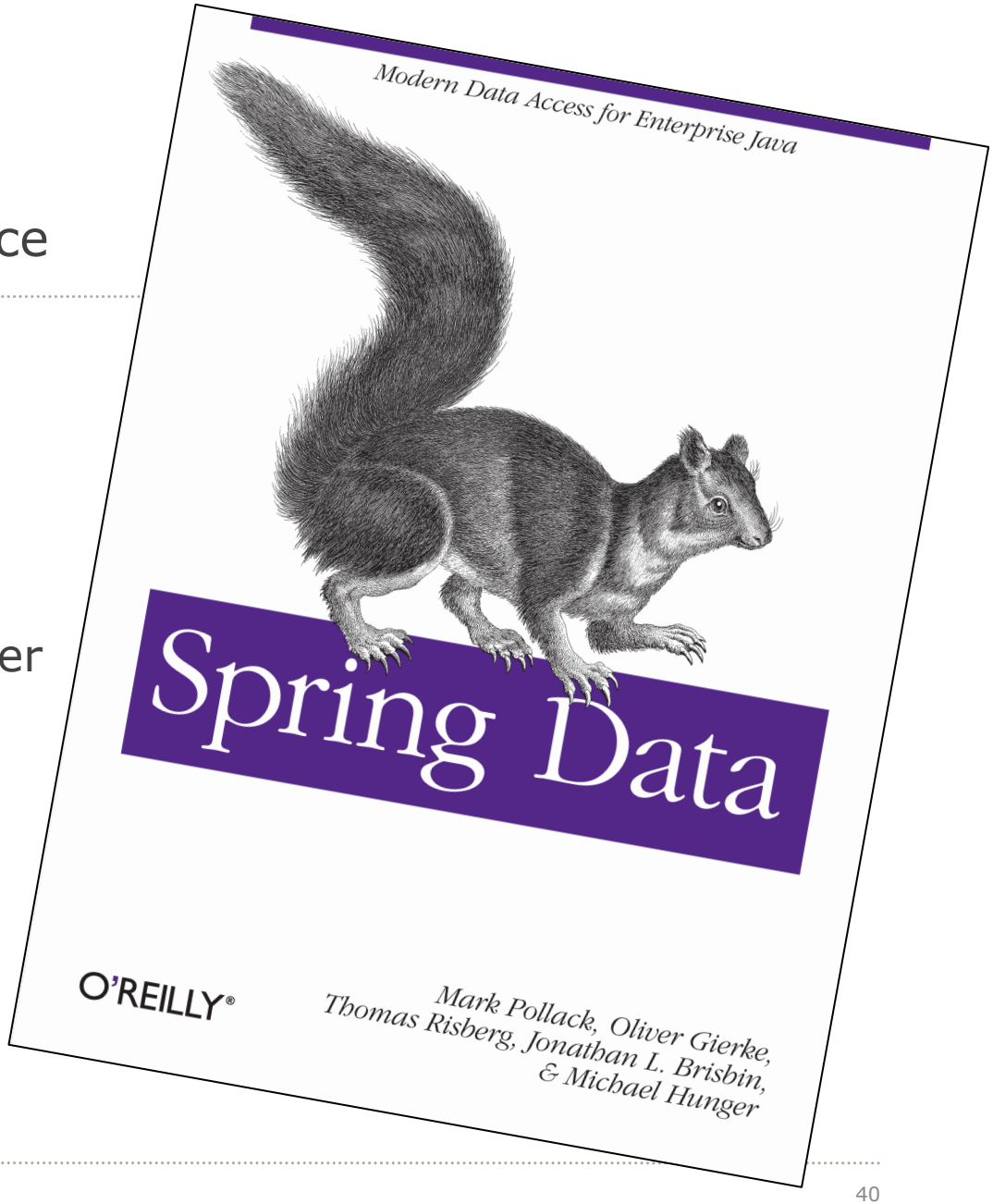
Splunk NoSQL Redis BigData

Hadoop Redis Roo Gemfire

MongoDB Neo4j REST exporter

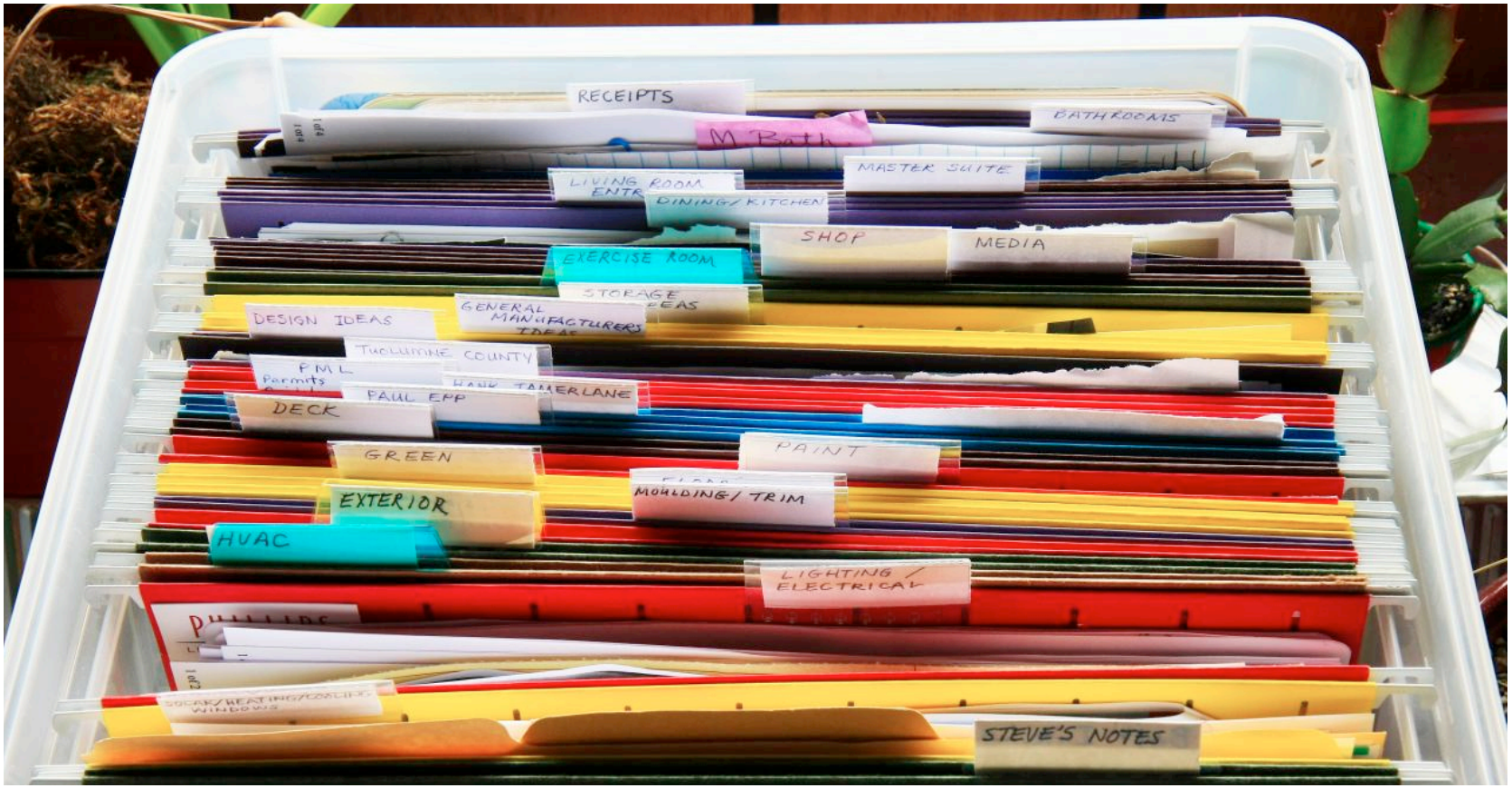
Querydsl Repositories

http://bit.ly/sd-book



Give it a try!

Use a data model which really matches to your data …



http://www.flickr.com/photos/juniorvelo/3267647833

Q&A

Thank you!

Patrick Baumgartner, @patbaumgartner
patrick.baumgartner [at] swiftmind [dot] com

http://www.swiftmind.com @swiftmind