

Introduction to Spring Data

Greg Turnquist
@gregturn
github.com/gregturn

Unless otherwise indicated, these slides are © 2013-2014 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: <http://creativecommons.org/licenses/by-nc/3.0/>

Team · Greg Turnquist

https://spring.io/team/gturnquist

Greg.L.Turnqui... !

spring

DOCS GUIDES PROJECTS BLOG QUESTIONS

Salt Lake City

United States
Greg Turnquist

Test-bitten script junky
Clarksville, TN USA

Greg is a test-bitten script junky. He is a member of the Spring team at Pivotal. He works on Spring Data REST, Spring Boot and other Spring projects, while also working as an editor-at-large of Spring's Getting Started guides. He launched the Nashville JUG in 2010. He created Spring Python and wrote "Spring Python 1.1" and "Python Testing Cookbook". He is currently writing "Learning Spring Boot".

[Twitter](#) [GitHub](#) [LinkedIn](#) [Medium](#)

GregLTurnquist.com/list



A screenshot of a web browser showing Greg Turnquist's profile on the Spring website. The browser window has a title bar "Team · Greg Turnquist" and a status bar "Greg.L.Turnqui...". The address bar shows the URL <https://spring.io/team/gturnquist>. The page features a dark header with the Spring logo and navigation links for DOCS, GUIDES, PROJECTS, BLOG, and QUESTIONS. Below the header is a map of the United States with a callout box highlighting Greg Turnquist's location in Clarksville, TN USA. A portrait photo of Greg Turnquist is displayed on the left. To the right of the map, there is a bio section with his name, title, and location, followed by a paragraph about his work and interests. Social media icons for Twitter, GitHub, LinkedIn, and a blog are shown. At the bottom right, there is a stack of three books by Greg L. Turnquist: "Learning Spring Boot", "Spring Python 1.1 Testing Cookbook", and "Spring Python 1.1 Programming".

Greg Turnquist
Test-bitten script junky
Clarksville, TN USA

Greg is a test-bitten script junky. He is a member of the Spring team at Pivotal. He works on Spring Data REST, Spring Boot, and various other Spring projects, while also working as an editor-at-large for the Spring Getting Started guides. He launched the Nashville JUG and has created Spring Python and wrote "Spring Python 1.1 Testing Cookbook". He is currently writing "Learning Spring Boot".

Learning Spring Boot

Community Experience Distilled

Learn how to use Spring Boot to build apps faster than ever before

Foreword by Dave Syer, Senior Engineering Consultant and Co-leader for Spring Boot

Greg L. Turnquist

[PACKT] open source

Greg L. Turnquist

[PACKT] open source

Greg L. Turnquist

[PACKT] open source

GregLTurnquist.com/list

pivotal

TM

DEVNEXUS™

 spring
by Pivotal

pivotal

TM

We're hiring!
(Atlanta + the world)
<http://pivotal.io/careers>

It's 2016

```
@Data  
@Entity  
class Employee {  
    @Id  
    @GeneratedValue  
    private Long id;  
    private String name;  
    private String role;  
  
    @ManyToOne  
    private Manager manager;  
  
    public Employee(String name, String role) {  
        this.name = name;  
        this.role = role;  
    }  
}
```

It's 2016

Why are we still writing...

It's 2016

Why are we still writing...

```
select * from EMPLOYEE  
where FIRST_NAME = %1
```

It's 2016

Why are we still writing...

```
select * from EMPLOYEE  
where FIRST_NAME = %1
```

```
select e from Employee e  
where e.firstName = :name
```

It's 2016

Why are we still writing...

```
select * from EMPLOYEE  
where FIRST_NAME = %1
```

```
select e from Employee e  
where e.firstName = :name
```

create

```
    .select()  
    .from(EMPLOYEE)  
    .where(EMPLOYEE.FIRST_NAME  
          .equal(name))  
    .fetch()
```

It's 2016

Why are we still writing...

SQL

```
select * from EMPLOYEE  
where FIRST_NAME = %1
```

JPA

```
select e from Employee e  
where e.firstName = :name
```

jOOQ

```
create  
    .select()  
    .from(EMPLOYEE)  
    .where(EMPLOYEE.FIRST_NAME  
          .equal(name))  
    .fetch()
```

When we could write...

```
interface EmployeeRepository extends CrudRepository<Employee, Long> {  
    List<Employee> findByFirstName(String firstName);  
}
```

When we could write...

```
interface EmployeeRepository extends CrudRepository<Employee, Long> {  
    List<Employee> findByFirstName(String firstName);  
}
```

...includes **save()**,
findOne(),
findAll()
exists(),
delete(),
and **count()**

10+ years ago

SQL

Today

SQL

NoSQL

Today

CQL

N1QL



SQL

NoSQL

Cypher

MongoDB CLI

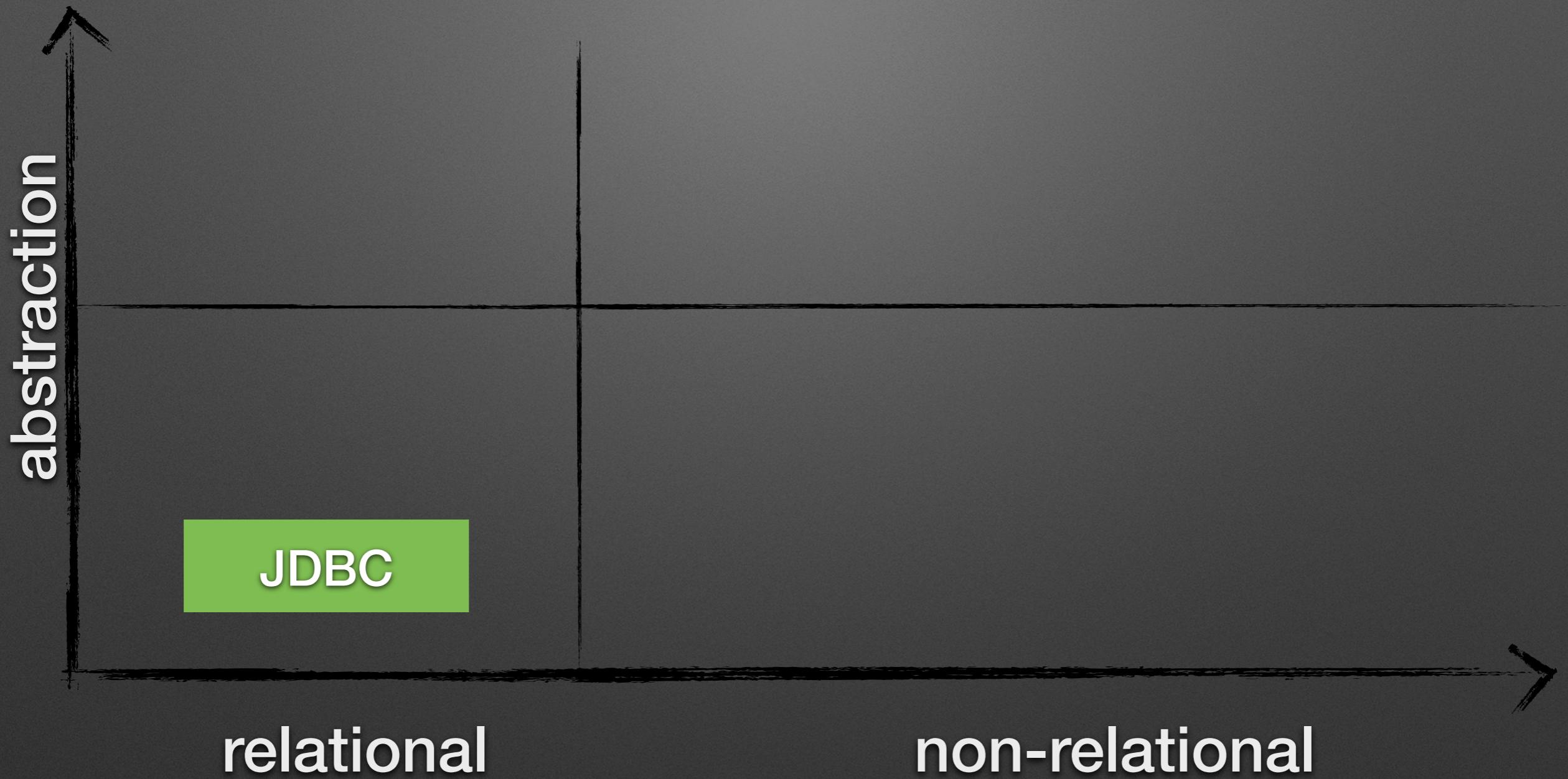
OQL

HQL

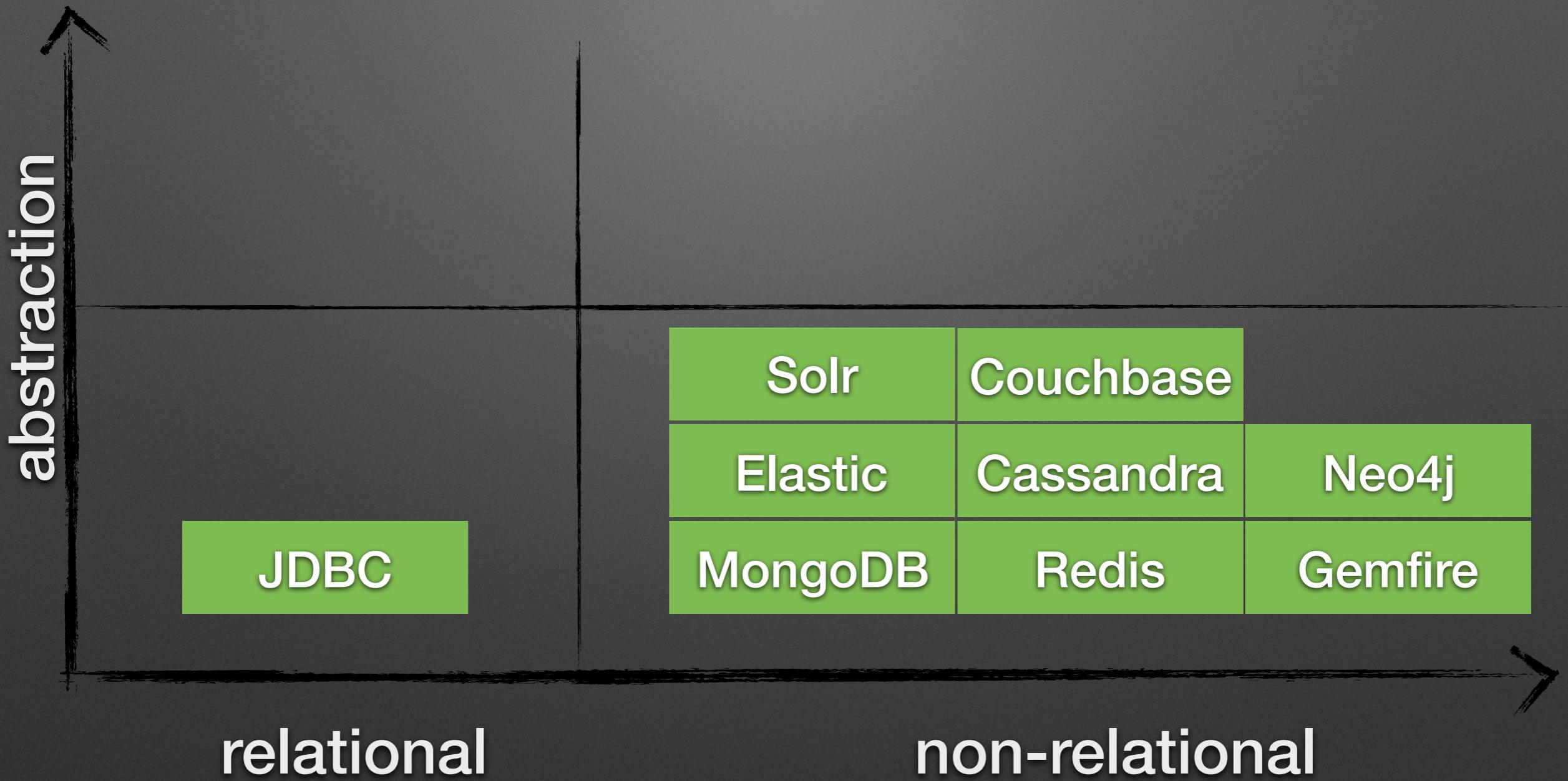
EJBQL



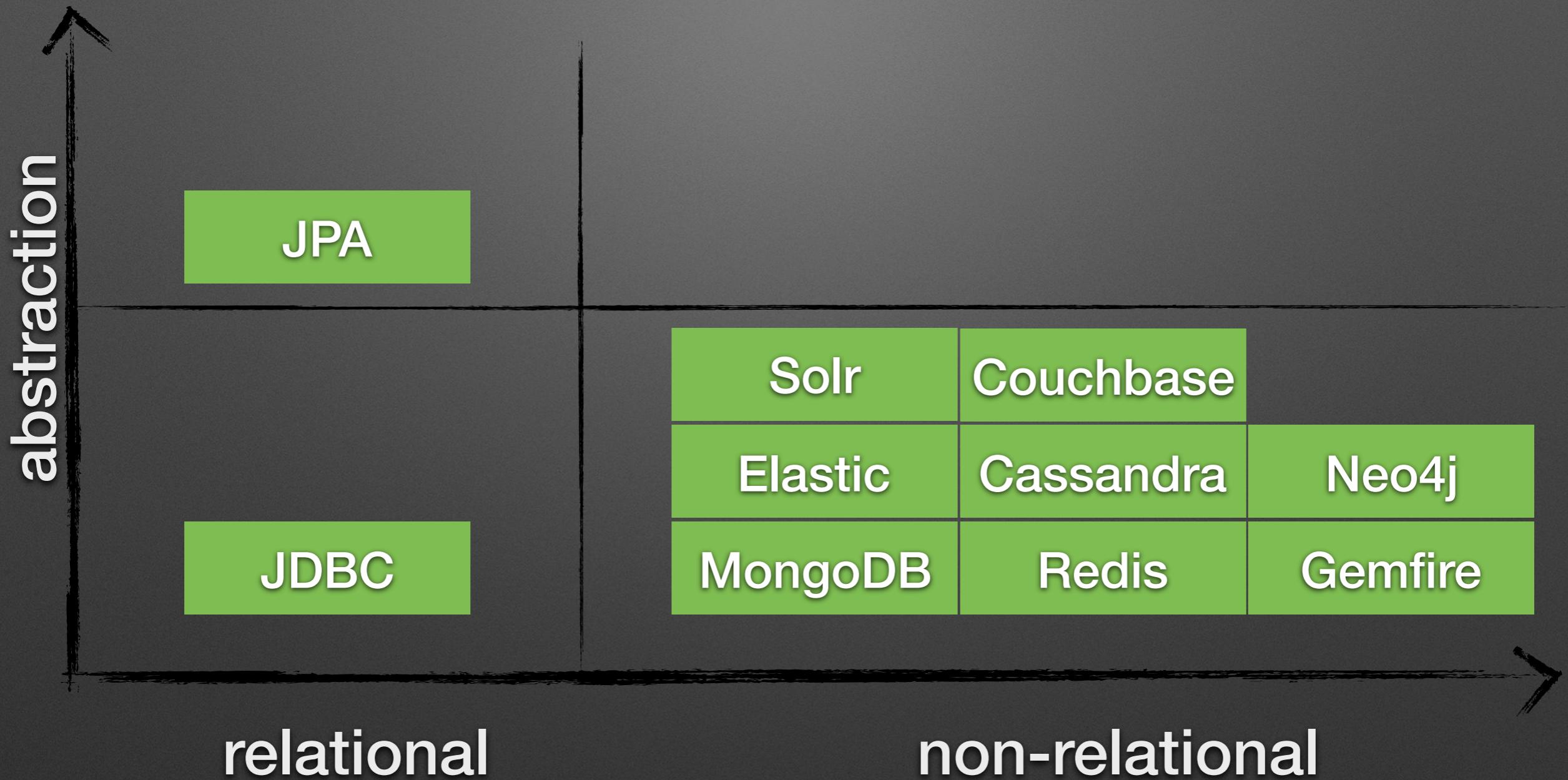
Spring Data provides...



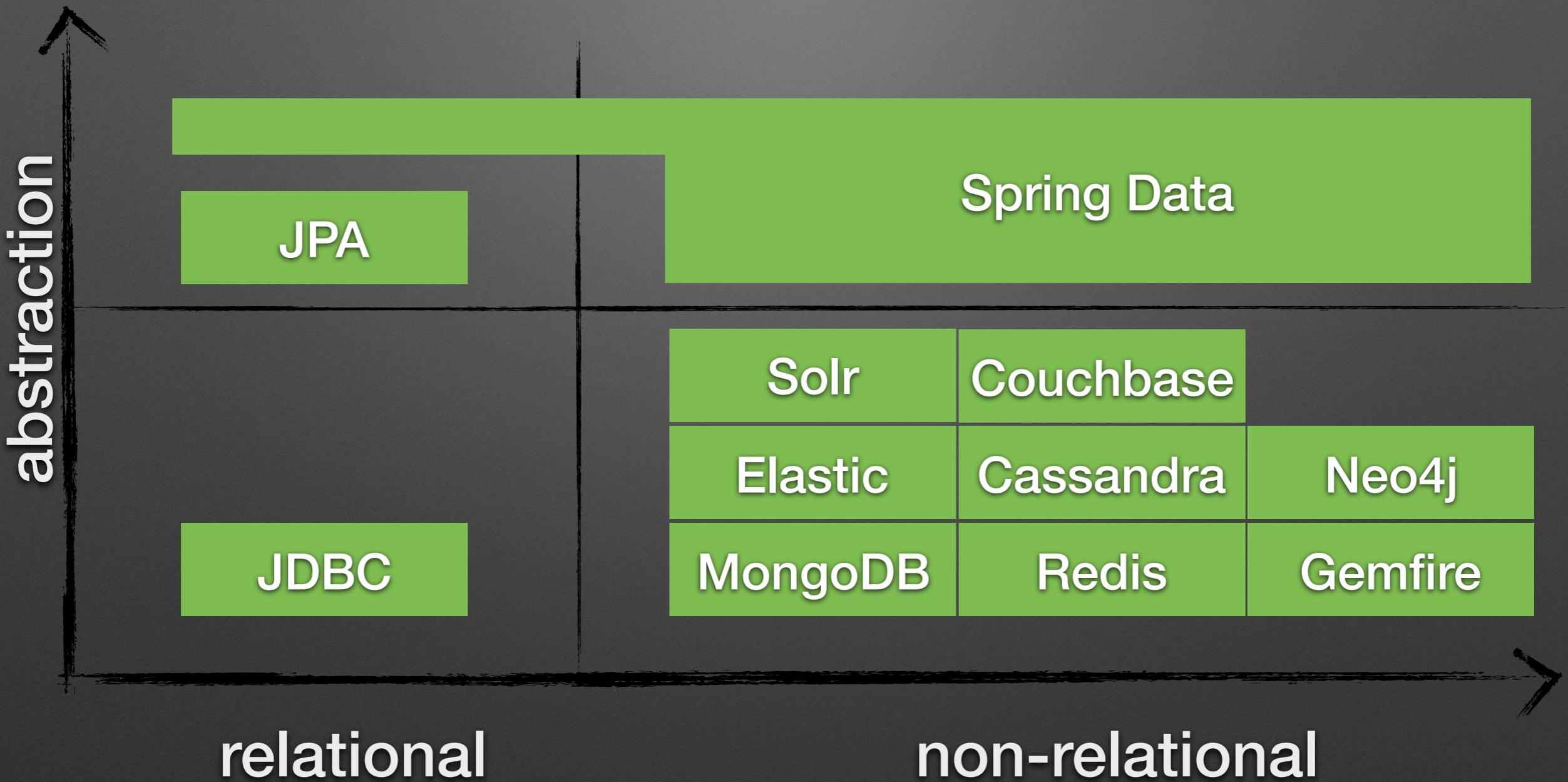
Spring Data provides...



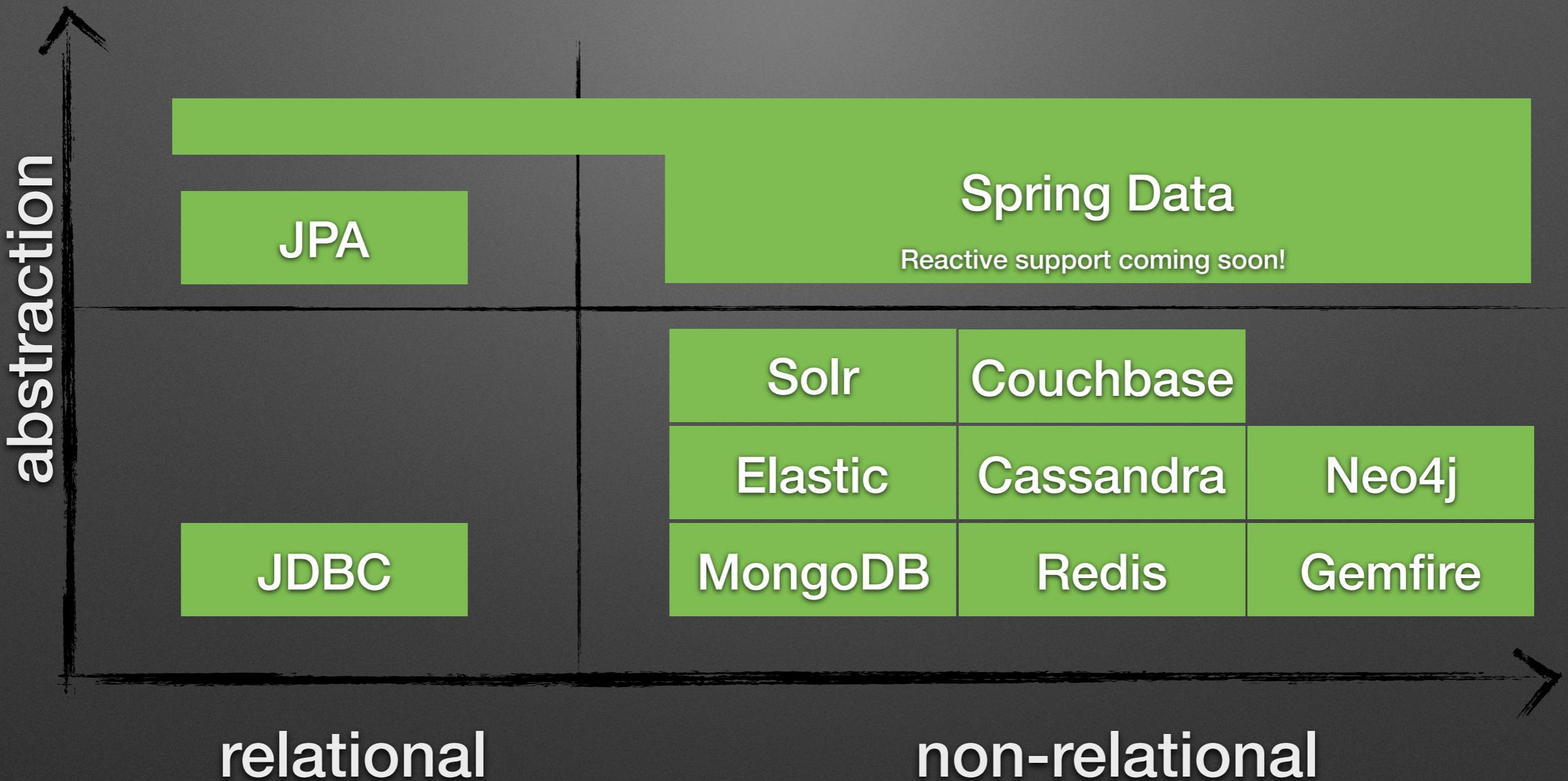
Spring Data provides...



Spring Data provides...



Spring Data provides...



Dialing it up

```
interface EmployeeRepository extends CrudRepository<Employee, Long> {  
  
    List<Employee> findByLastName(String f)  
    List<Employee> findByFirstNameAndLastName(String f, String l)  
    List<Employee> findByFirstNameAndManagerName(String f, String m)  
    List<Employee> findTop10ByFirstName(...) // or findFirst10ByFirstName  
    List<Employee> findDistinctEmployeesByFirstName(...)  
    List<Employee> findByFirstNameAndLastNameAllIgnoreCase(...)  
    List<Employee> findByFirstNameOrderByLastNameAsc(...)  
    List<Employee> findByLastNamesNull(...)  
}
```

Why stop there?

```
interface CoolRepo extends CrudRepository<Employee, Long> {  
    Stream<Employee> findByLastname(String lastname)  
    @Async Future<Employee> findByLastname(...)  
    @Async CompletableFuture<Employee> findByLastname(...)  
    @Async ListenableFuture<Employee> findByLastname(...)  
  
    Page<Employee> findAll(Pageable p)  
    Page<Employee> findByFirstName(String f, Pageable p)  
    List<Employee> findAll(Sort s)  
    Page<Employee> findByFirstName(String f, Pageable p, Sort s)  
}
```

Why stop there?

```
interface CoolRepo extends CrudRepository<Employee, Long> {  
    Stream<Employee> findByLastname(String lastname)  
    @Async Future<Employee> findByLastname(...)  
    @Async CompletableFuture<Employee> findByLastname(...)  
    @Async ListenableFuture<Employee> findByLastname(...)  
  
    Page<Employee> findAll(Pageable p)  
    Page<Employee> findByFirstName(String f, Pageable p)  
    List<Employee> findAll(Sort s)  
    Page<Employee> findByFirstName(String f, Pageable p, Sort s)  
}
```

Use `@Query` to write custom queries

Spring Data Demo

Unless otherwise indicated, these slides are
© 2013-2014 Pivotal Software, Inc. and
licensed under a Creative Commons Attribution-
NonCommercial license: <http://creativecommons.org/licenses/by-nc/3.0/>

Links

- <http://projects.spring.io/spring-data>
- <https://github.com/spring-projects/spring-data-examples>
- <https://spring.io/guides/tutorials/react-and-spring-data-rest/>
- <https://spring.io/guides?filter=spring%20data>

Introduction to Spring Data

Greg Turnquist
@gregturn
github.com/gregturn

Unless otherwise indicated, these slides are © 2013-2014 Pivotal Software, Inc. and licensed under a Creative Commons Attribution-NonCommercial license: <http://creativecommons.org/licenses/by-nc/3.0/>