RoR + DynamoDB = ?

Silvia Schreier, @aivlis_s

JRubyConf 2013, Berlin

innoQ

What we wanted

RESTful and ROCA-compliant[1] web application

Cloud deployment using AWS

High scalability

=› Ruby on Rails + DynamoDB

[1] http://roca-style.org

DynamoDB in a nutshell

What is DynamoDB?

"fully managed NoSQL database service" [1]

Variable scaling

API and Web based console

SDKs for multiple languages

Combination with other AWS services

[1] http://aws.amazon.com/dynamodb/

Tables

2 types with different kind of keys

Get, (partial) update, delete

Batch operations

Write operations support preconditions

No key generation

Tables with hash key

ID string	name string	year of birth number	biography binary
“ab12”	“Daisy Duck”	1975	
“xy56”	“Micky Mouse”		long compressed text
“op13”	“Goofy”	1968	
“fx81”	“Donald Duck”	1970	

Tables with hash & range key

authorID string	bookID number	title string	reviews string set
“xy56”	19991234	“Duckburg Tales”	[“nice one”, “like it”]
“xy56”	20123323	“Mouse & More”	
“fx81”	20105453	“Money, Money, Money”	[“not my favourite”]
“op13”	20118963	“My Life”	

Query vs. Scan[1]

Both support paging and limiting

Query is more efficient

Query works only on tables with range key

Scan examines every item

Avoid scan especially on large tables

Scan can be parallelized

[1] http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/QueryAndScan.html

Persistence Module

Ruby, Rails and SDK

Ruby 2 & Rails 4

experimental 2.0 support in JRuby (since 1.7.4)

AWS Ruby SDK needs 1.7.5 (see #437[1])

Low- and high-level access to DynamoDB

[1] https://github.com/jruby/jruby/issues/437

What does it look like? (I)

```
class Author

  include PersistableWithHash

  self.table_name    = 'authors'
  self.hash_key_name = 'id'

  self.enable_revisioning

  with_attributes :name, :id
  add_attribute :year_of_birth, as: :number

  def initialize(attributes = {})
    super(attributes)
  end

  def create_id
    self.id = SecureRandom.uuid
  end

end
```

What does it look like? (I)

```
class Author

  include Persi

  self.table_na
  self.hash_key

  self.enable_r

  with_attribut
  add_attribute

  def initializ
    super(attributes)
  end

  def create_id
    self.id = SecureRandom.uuid
  end

end
```

```
irb> a = Author.new
irb> a.name = 'Donald Duck'
irb> a.save
irb> id = a.id
irb> found = Author.find_by_hash id
irb> found.name
=> 'Donald Duck'
irb>
```

What does it look like? (II)

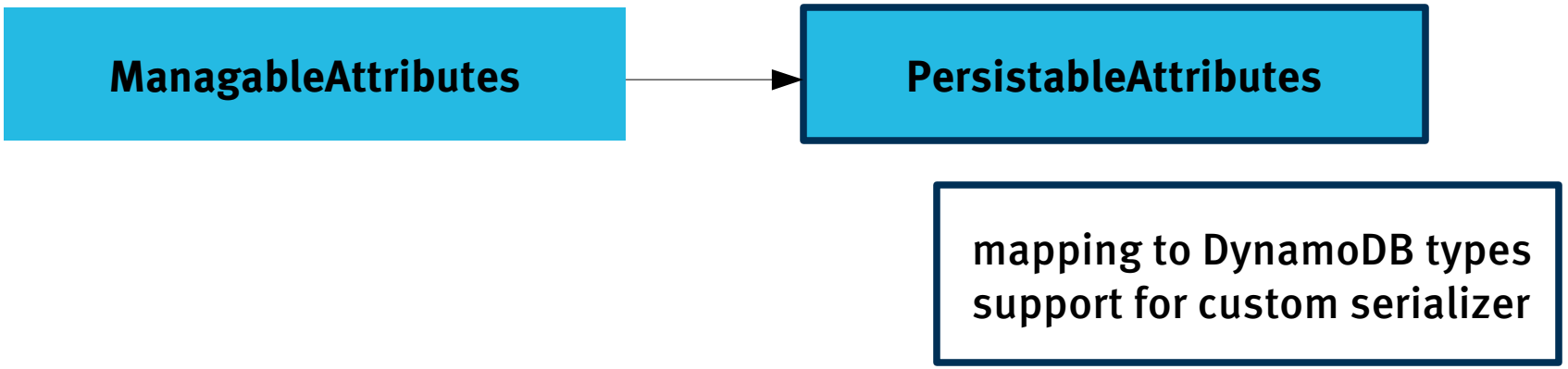
```
class Book

  include PersistableWithRange

  self.table_name      = 'books'
  self.hash_key_name   = 'author_id'
  self.range_key_name = 'book_id'

  with_attributes :author_id, :title
  add_attribute :book_id, as: :number
  add_attribute :reviews, as: :string, collection: true

  add_attribute_index :title, ascending: true

  set_serializer :reviews, Review

  # ...

end
```

Persistence Module Design

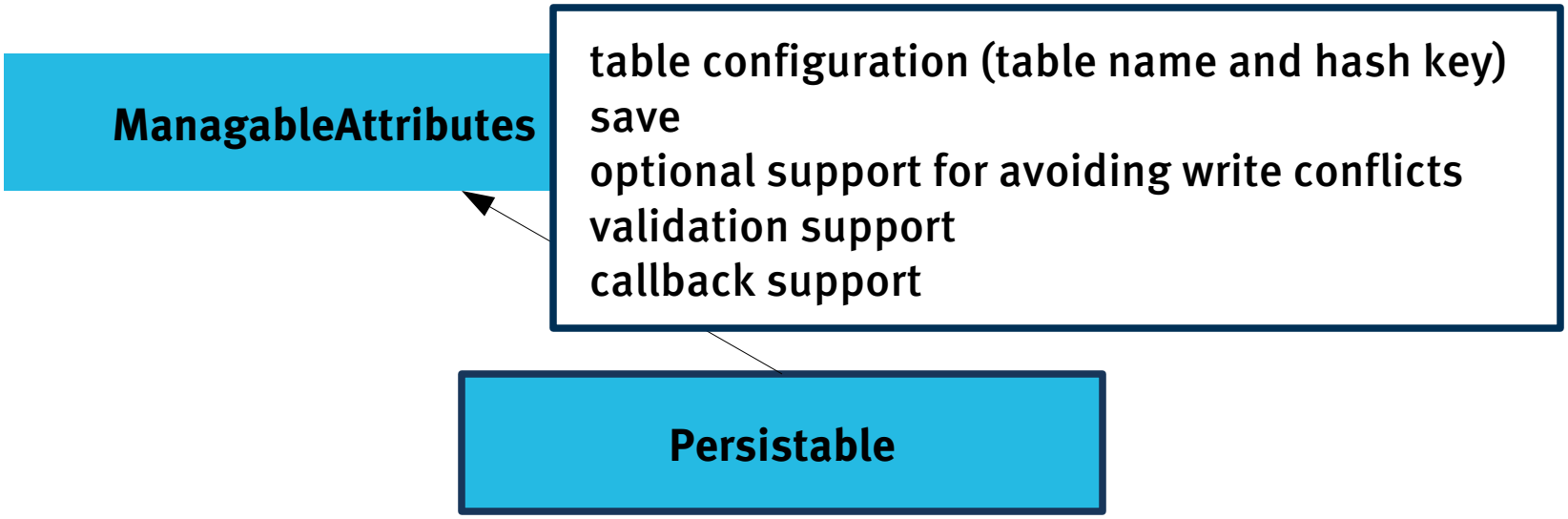
ManagableAttributes

typed attributes (string (set), number (set))
stored in a hash
generation of accessors

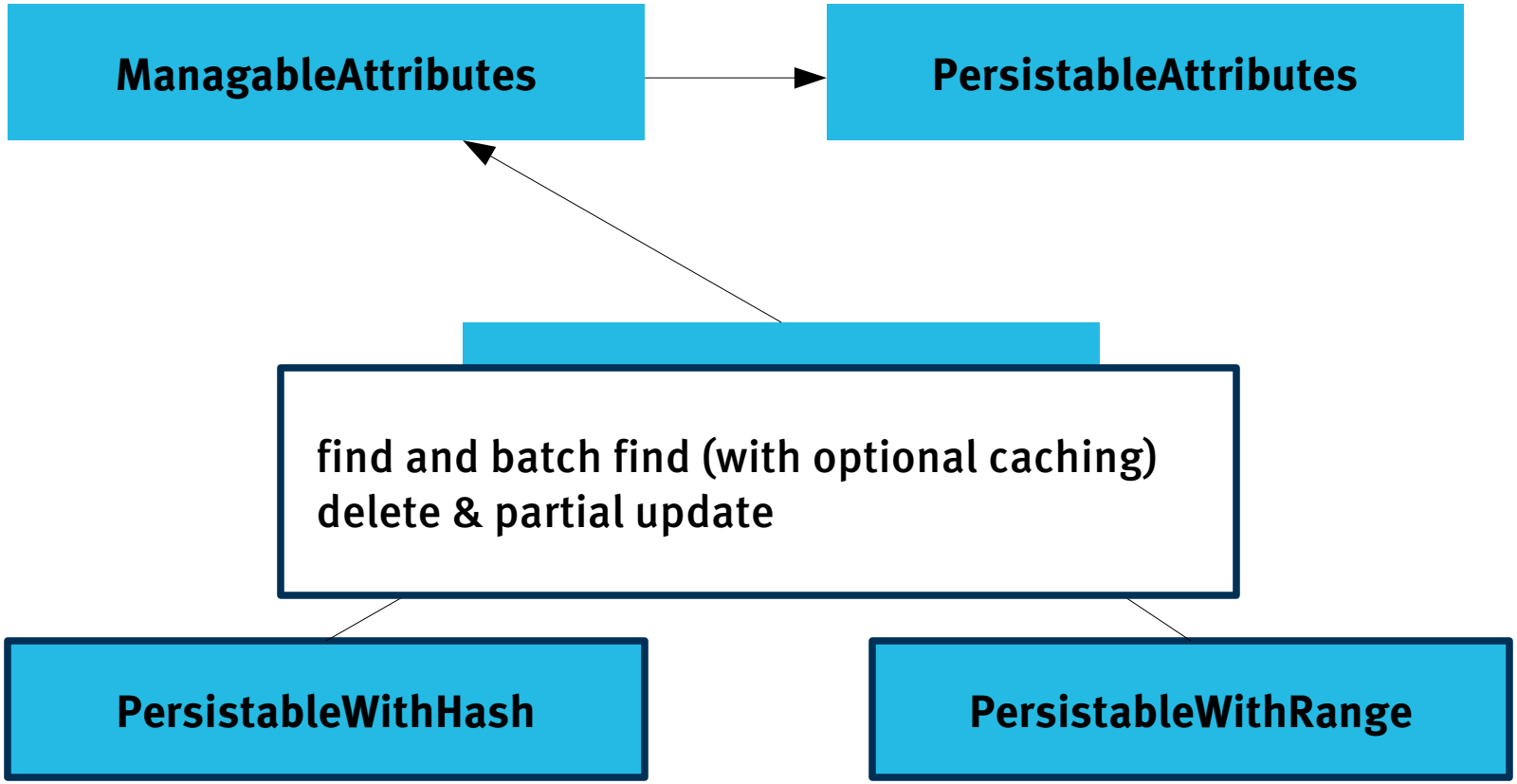
Persistence Module Design



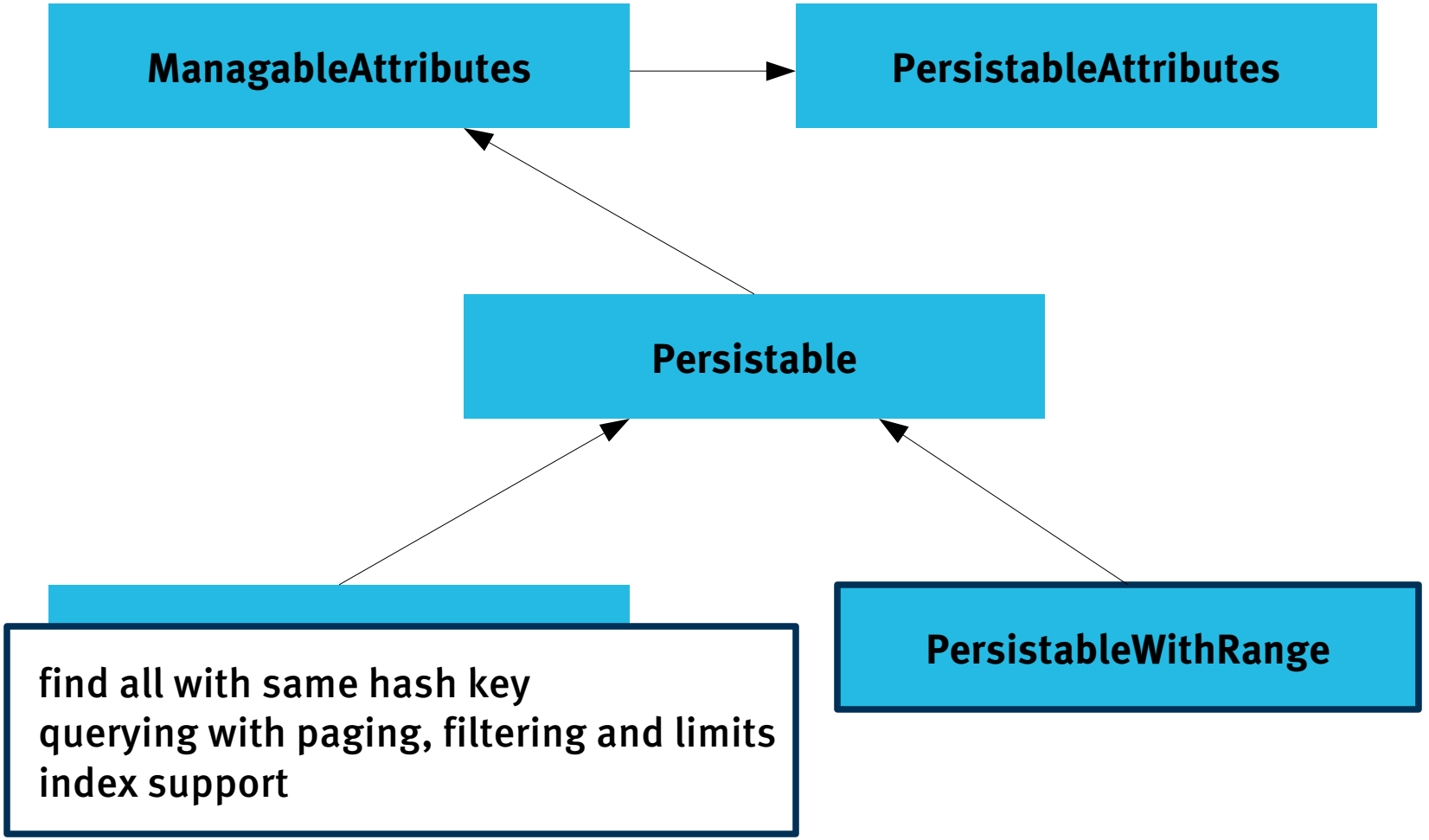
Persistence Module Design



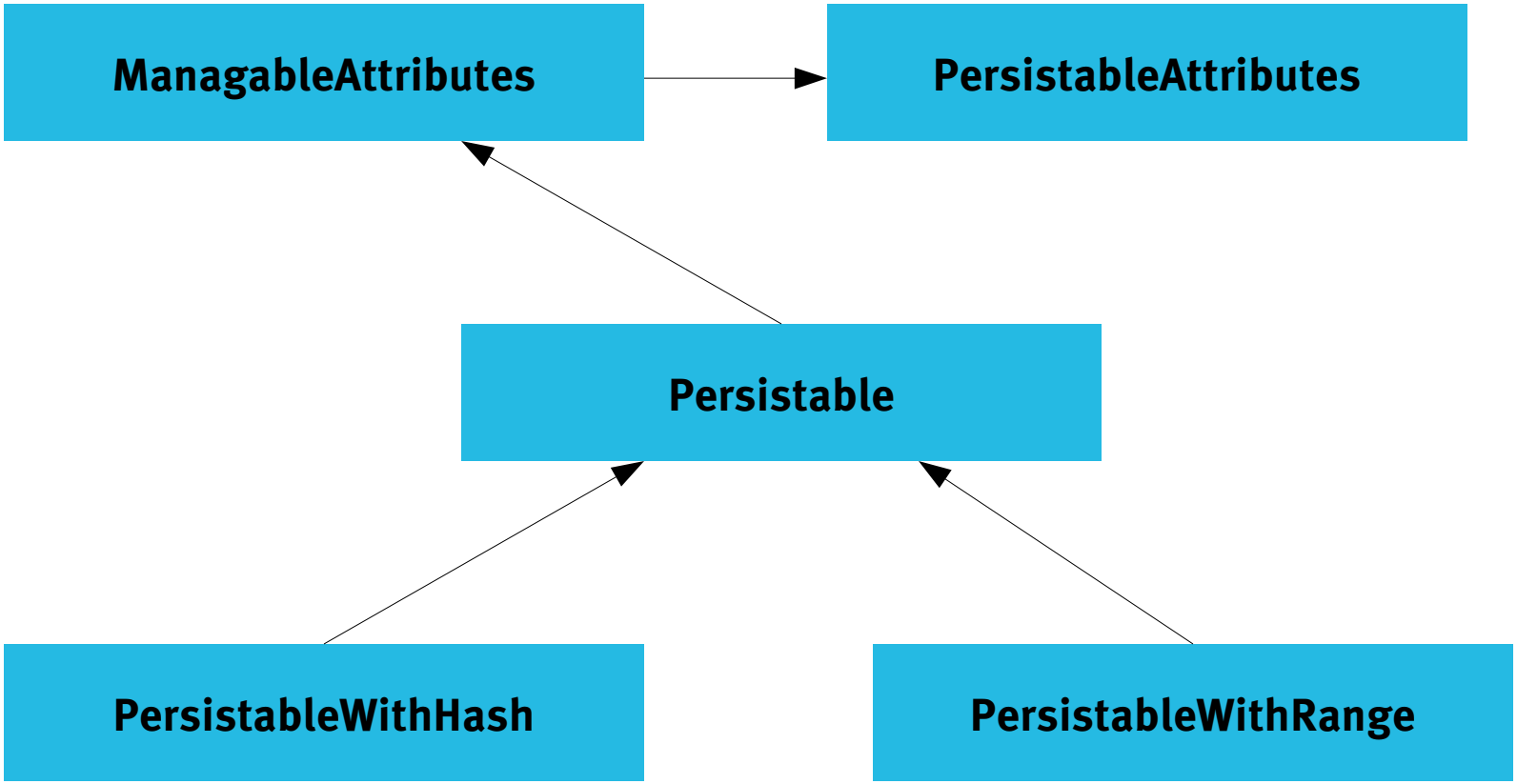
Persistence Module Design



Persistence Module Design



Persistence Module Design



What next?

Adding scan support

Monitor performance & performance tests

Automated scaling[1]

Database mock for offline testing

[1] https://github.com/invisiblehand/dynamo-autoscale

Lessons learned

Easy setup

Automate table management

Different design thinking (Keys are the key!)

Follow best practices[1]

Limiting and Paging not optimal

[1] http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/BestPractices.html

Thank you for your attention!

Questions?

Silvia Schreier, @aivlis_s
silvia.schreier@innoq.com
http://www.innoq.com

innoQ