How to use Arel?

Wrong answers only!

dhalasz@redhat.com

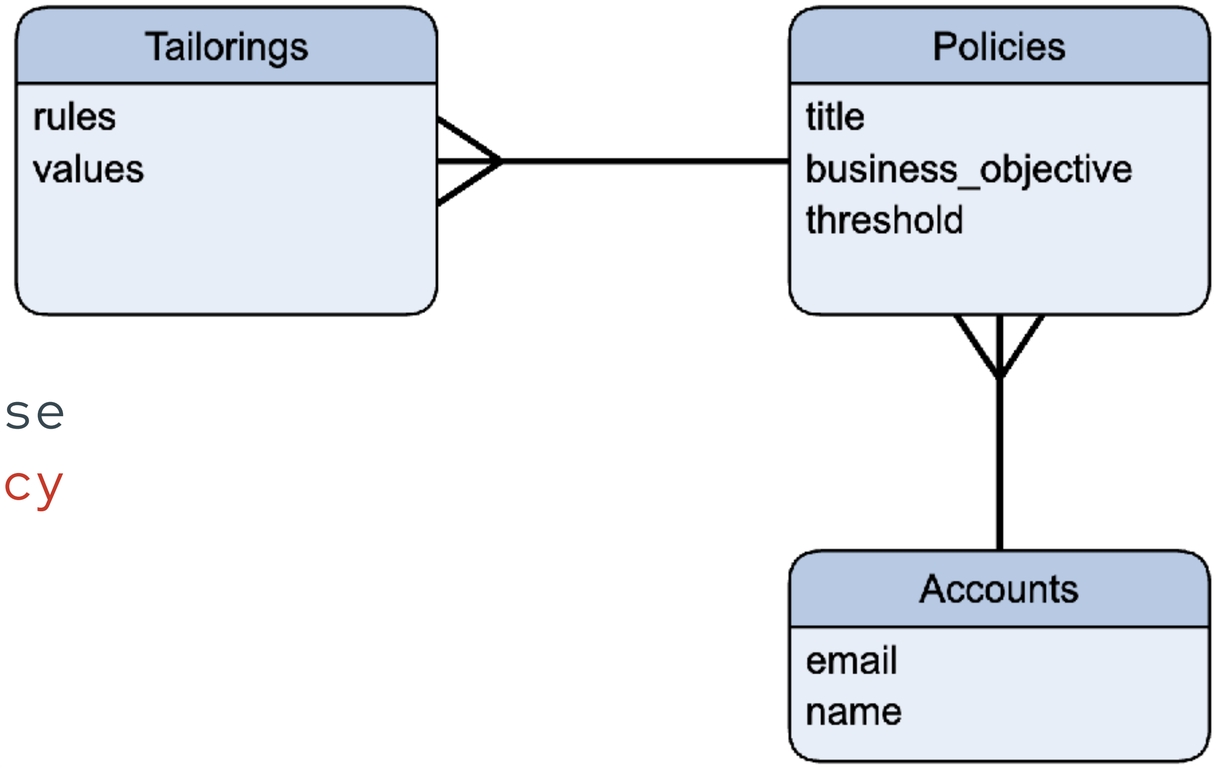
Dávid Halász

www.skateman.eu

```ruby
class Policy < ActiveRecord::Base
  has_many :tailorings
  belongs_to :account
end

class Tailoring < ActiveRecord::Base
  has_one :account, through: :policy
  belongs_to :policy
end

class Account < ActiveRecord::Base
  has_many :policies
  has_many :tailorings, through: :policies
end
```



```
Tailoring.joins(:policy)
```

```
Tailoring.joins(:policy)
```

```
SELECT "tailorings".* FROM "tailorings"
INNER JOIN "policies" ON "policies"."id" = "tailorings"."policy_id"
```

```ruby
Tailoring.joins(:policy).where('policies.title = "foo"')
```

```
Tailoring.joins(:policy).where('policies.title = "foo"')
```

*Or you can just use Arel

```
Tailoring.joins(:policy).where(policy: {title: 'foo'})
```

```ruby
Tailoring.joins(:policy).where(policy: {title: 'foo'})
```

```sql
SELECT "tailorings".* FROM "tailorings"
INNER JOIN "policies" “policy"
ON "policy"."id" = "tailorings"."policy_id"
WHERE "policy"."title" = 'foo'
```

```ruby
Tailoring.joins(:policy).where(policy: {title: 'foo'})
```

```sql
SELECT "tailorings".* FROM "tailorings"
INNER JOIN "policies" "policy"
ON "policy"."id" = "tailorings"."policy_id"
WHERE "policy"."title" = 'foo'
```

Rails automatically creates an alias

Except when it doesn’t

```ruby
class Policy < ActiveRecord::Base
  has_many :tailorings
  belongs_to :account



end

class Tailoring < ActiveRecord::Base
  has_one :account, through: :policy
  belongs_to :policy
end

class Account < ActiveRecord::Base
  has_many :policies
  has_many :tailorings, through: :policies
end
```

```ruby
class Policy < ActiveRecord::Base
  has_many :tailorings
  belongs_to :account

  scope :strict, -> { where(threshold: 100) }
end

class Tailoring < ActiveRecord::Base
  has_one :account, through: :policy
  belongs_to :policy
end

class Account < ActiveRecord::Base
  has_many :policies
  has_many :tailorings, through: :policies
end
```

```
Tailoring.joins(:policy).where(policy: {
  title: 'foo',
  id: Policy.strict.select(:id)
})
```

```ruby
Tailoring.joins(:policy).where(policy: {
  title: 'foo',
  id: Policy.strict.select(:id)
})
```

```sql
SELECT "tailorings".* FROM "tailorings"
INNER JOIN "policies" "policy"
ON "policy"."id" = "tailorings"."policy_id"
WHERE "policy"."title" = 'foo' AND "policy"."id" IN (
  SELECT "policies"."id" FROM "policies"
  WHERE "policies"."threshold" = 100
)
```

*Or you can just use Arel

ActiveRecord::SpawnMethods#merge

```
Tailoring.joins(:policy).where(policy: {title: 'foo'})
                        .merge(Policy.strict)
```

```ruby
Tailoring.joins(:policy).where(policy: {title: 'foo'})
                        .merge(Policy.strict)
```

```sql
SELECT "tailorings".* FROM "tailorings"
INNER JOIN "policies" "policy"
ON "policy"."id" = "tailorings"."policy_id"
WHERE "policy"."title" = 'foo' AND "policies"."threshold" = 100
```

It’s nice right?

Except it doesn’t work

```
Tailoring.joins(:policy).where(policy: {title: 'foo'})
                        .merge(Policy.strict)
```

```
SELECT "tailorings".* FROM "tailorings"
INNER JOIN "policies" "policy"
ON "policy"."id" = "tailorings"."policy_id"
WHERE "policy"."title" = 'foo' AND "policies"."threshold" = 100
```

Rails automatically creates an alias

Except when it doesn’t

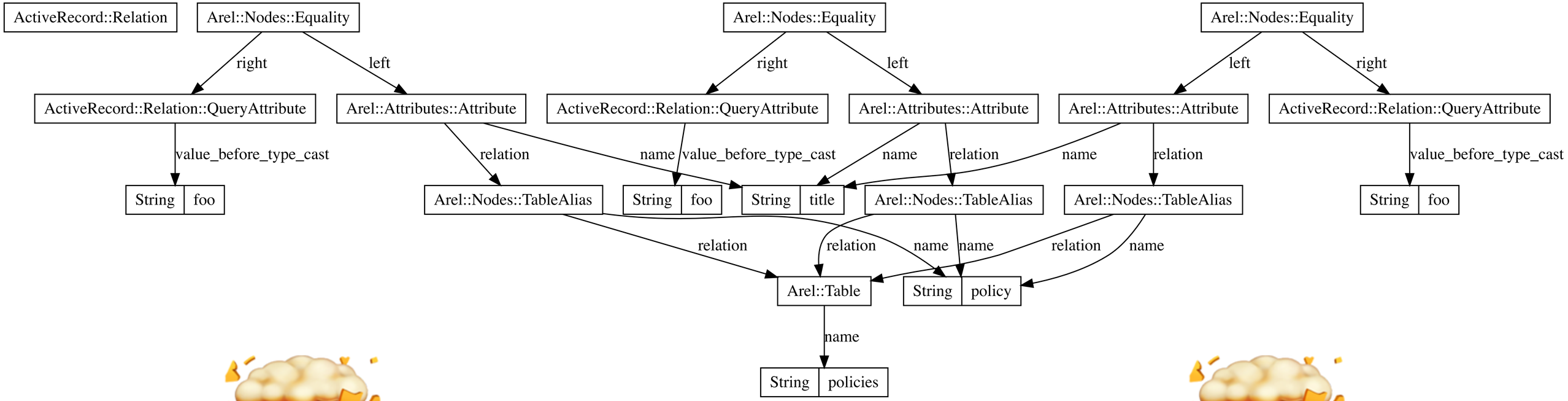
```
Tailoring.joins(:policy).where(policy: {title: 'foo'})
                        .merge(Policy.strict)
```

*Or you can just use Arel

What happens under the hood?

```
Tailoring.joins(:policy).where(policy: { title: 'foo'})
                        .where_clause.ast
```

```
#<Arel::Nodes::Equality:0x0000ffff84252018
 @left=
  #<struct Arel::Attributes::Attribute
   relation=
    #<Arel::Nodes::TableAlias:0x0000ffff842522e8
     @left=
      #<Arel::Table:0x0000ffff84e91978
       @klass=Policy(id: uuid, title: string, threshold: float, business_objective: string, profile_id:
uuid, account_id: uuid),
       @name="policies",
       @table_alias=nil,
       @type_caster=
        #<ActiveRecord::TypeCaster::Map:0x0000ffff84e91900
         @klass=Policy(id: uuid, title: string, threshold: float, business_objective: string, profile_id:
uuid, account_id: uuid)>>,
     @right="policy">,
   name="title">,
 @right=
  #<ActiveRecord::Relation::QueryAttribute:0x0000ffff84252068
   @name="title",
   @original_attribute=nil,
   @type=#<ActiveModel::Type::String:0x0000ffff84e1f8c8 @false="f", @limit=nil, @precision=nil,
@scale=nil, @true="t">,
   @value_before_type_cast="foo">>
```


ActiveRecord::Relation
Arel::Nodes::Equality
right
left
ActiveRecord::Relation::QueryAttribute
Arel::Attributes::Attribute
value_before_type_cast
String
foo
relation
name
Arel::Nodes::TableAlias
String
foo
String
title
Arel::Nodes::Equality
right
left
ActiveRecord::Relation::QueryAttribute
Arel::Attributes::Attribute
value_before_type_cast
name
relation
Arel::Nodes::TableAlias
relation
name
name
relation
Arel::Table
String
policy
name
String
policies
Arel::Nodes::Equality
left
right
Arel::Attributes::Attribute
ActiveRecord::Relation::QueryAttribute
name
relation
Arel::Nodes::TableAlias
relation
name
value_before_type_cast
String
foo

```
#<Arel::Nodes::Equality:0x0000ffff84252018
 @left=
  #<struct Arel::Attributes::Attribute
   relation=
    #<Arel::Nodes::TableAlias:0x0000ffff842522e8
     @left=
      #<Arel::Table:0x0000ffff84e91978
       @klass=Policy(id: uuid, title: string, threshold: float, business_objective: string, profile_id:
uuid, account_id: uuid),
       @name="policies",
       @table_alias=nil,
       @type_caster=
        #<ActiveRecord::TypeCaster::Map:0x0000ffff84e91900
         @klass=Policy(id: uuid, title: string, threshold: float, business_objective: string, profile_id:
uuid, account_id: uuid)>>,
     @right="policy">,
   name="title">,
 @right=
  #<ActiveRecord::Relation::QueryAttribute:0x0000ffff84252068
   @name="title",
   @original_attribute=nil,
   @type=#<ActiveModel::Type::String:0x0000ffff84e1f8c8 @false="f", @limit=nil, @precision=nil,
@scale=nil, @true="t">,
   @value_before_type_cast="foo">>
```

Arel::Visitor

For traversing Arel expressions

*and generating raw SQL

```
Tailoring.joins(:policy).where(policy: {title: 'foo'})
```

```
aliases = {}

visitor = ArelVisitor.new do |node|
  if node.is_a?(Arel::Nodes::TableAlias)
    aliases[node.table_name] = node.name
  end
end

visitor.accept(where_clause.ast)
```

Policy.strict

```
visitor = ArelVisitor.new(copy: true) do |node|
  if node.is_a?(Arel::Table) && aliases.key?(node.name)
    node.alias(aliases[node.name])
  elsif node.is_a?(Arel::Nodes::TableAlias) && aliases.key?(node.left.name)
    node.left.alias(aliases[node.left.name])
  else
    node
  end
end

right_side = visitor.accept(where_clause.ast)
```

```
Tailoring.joins(:policy).where(policy: {title: 'foo'})
                        .where(right_side)
```

```ruby
Tailoring.joins(:policy).where(policy: {title: 'foo'})
                        .where(right_side)
```

```sql
SELECT "tailorings".* FROM "tailorings"
INNER JOIN "policies" "policy"
ON "policy"."id" = "tailorings"."policy_id"
WHERE "policy"."title" = 'foo' AND "policy"."threshold" = 100
```

*Or you can just use Arel

Thank you!

dhalasz@redhat.com

Dávid Halász

www.skateman.eu