Сложные SQL запросы в Ruby On Rails

Манылов Павел aka [r-k]
SFD 2012 / Екатеринбург

Сложные запросы?

- Не описаны в туториалах и документации по Rails
- Чаще всего затрагивают больше одной таблицы
- Вызывают затруднения при написании

Короткий обзор

- **Rails** - MVC фреймворк
- **ActiveRecord** - ORM библиотека
- **ARel** - менеджер синтаксических деревьев для составления sql-запросов

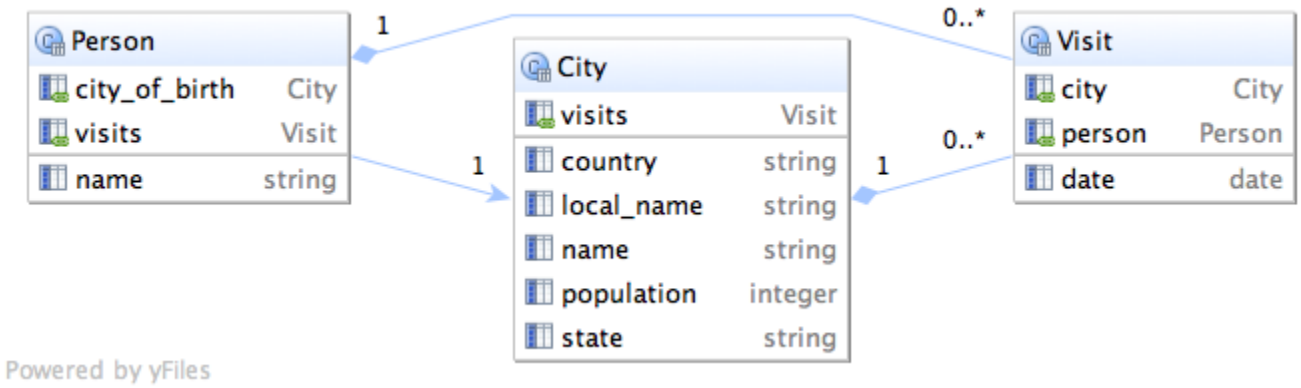
Rails использует **ActiveRecord** использует **ARel**

Требования к коду

- Красивость
- Читабельность
- Без вставок SQL-кода
- Гибкость, DRY
- Rails Way

Группировка и агрегирование

Используем следующую структуры БД



Группировка и агрегирование

Необходимо получить *население по областям (штатам)* в виде *массива объектов со следующими данными*:

```
[{
   :country=>"Страна",
   :state=>"Область",
   :population=>20000
 },
...]
```

Группировка и агрегирование

На чистом SQL:

```
City.find_by_sql("
  SELECT `country`,
         `state`,
          sum(`population`) as `population`
  FROM `cities`
  GROUP BY `country`, `state`
")
```

Группировка и агрегирование

На чистом SQL:

```
City.find_by_sql("
  SELECT `country`,
         `state`,
          sum(`population`) as `population`
  FROM `cities`
  GROUP BY `country`, `state`
")
```

Группировка и агрегирование

На чистом SQL:

```
City.find_by_sql("
  SELECT `country`,
         `state`,
          sum(`population`) as `population`
  FROM `cities`
  GROUP BY `country`, `state`
")
```

Группировка и агрегирование

ActiveRecord:

```
City.group(:country,:state) \
    .sum(:population)
```

Группировка и агрегирование

ActiveRecord:

```
City.group(:country,:state) \
    .sum(:population)
# => { ["Страна", "Область"] => 10000}
```

Группировка и агрегирование

ActiveRecord:

```
City.group(:country,:state) \
     .sum(:population) \
     .map{|k,v|
        {
            :country=>k[0],
            :state=>k[1],
            :population=>v
         }
      } # => Array<Hash>
```

Группировка и агрегирование

ActiveRecord:

```
City.group(:country,:state) \
    .sum(:population) \
    .map{|k,v|
        {
            :country=>k[0],
            :state=>k[1],
            :population=>v
        }
    } # => Array<Hash>
```

Группировка и агрегирование

ActiveRecord:

```
City.select(
    [
      :country,
      :state,
      "sum(`population`) as `population`"
    ])\
    .group(:country, :state)
```

Группировка и агрегирование

ActiveRecord:

```
City.select(
    [
      :country,
      :state,
      "sum(`population`) as `population`"
    ])\
    .group(:country, :state)
```

Группировка и агрегирование

ActiveRecord:

```
City.select(
    [
      :country,
      :state,
      "sum(`population`) as `population`"
    ])\
    .group(:country, :state)
```

Группировка и агрегирование

ActiveRecord:

```
City.select(
    [
      :country,
      :state,
      "sum(`population`) as `population`"
    ])\
    .group(:country, :state)
# => ActiveRecord::Relation
```

Группировка и агрегирование

ActiveRecord:

```
City.select(
    [
      :country,
      :state,
      "sum(`population`) as `population`"
    ])\
    .group(:country, :state)
```

Группировка и агрегирование

SQL by ARel:

```
at = City.arel_table
City.find_by_sql(
   at.project(
      at[:country],
      at[:state], at[:population].sum.as
      ("population")
   ).group(at[:country],at[:state])
)
```

Группировка и агрегирование

SQL by ARel:

```
at = City.arel_table
City.find_by_sql(
   at.project(
      at[:country],
      at[:state], at[:population].sum.as
      ("population")
   ).group(at[:country],at[:state])
)
```

Группировка и агрегирование

SQL by ARel:

```
at = City.arel_table
City.find_by_sql(
   at.project( # Arel::SelectManager.new(…).from(…).project(…)
      at[:country],
      at[:state], at[:population].sum.as
      ("population")
   ).group(at[:country],at[:state])
)
```

Группировка и агрегирование

SQL by ARel:

```
at = City.arel_table
City.find_by_sql(
   at.project(
      at[:country],
      at[:state], at[:population].sum.as
      ("population")
   ).group(at[:country],at[:state])
)
```

Группировка и агрегирование

SQL by ARel:

```
at = City.arel_table
City.find_by_sql(
   at.project(
      at[:country],
      at[:state], at[:population].sum.as
      ("population")
   ).group(at[:country],at[:state])
)
```

Группировка и агрегирование

SQL by ARel:

```
at = City.arel_table
City.find_by_sql(
   at.project(
      at[:country],
      at[:state], at[:population].sum.as
      ("population")
   ).group(at[:country],at[:state])
)
```

Группировка и агрегирование

ActiveRecord + ARel:

```
at = City.arel_table
City.select(
   [
      :country,
      :state, at[:population].sum.as
      ("population")
   ]
).group(:country, :state)
```

Группировка и агрегирование

ActiveRecord + ARel:

```
at = City.arel_table
City.select(
   [
      :country,
      :state, at[:population].sum.as
      ("population")
   ]
).group(:country, :state)
```

Вложенные запросы

Необходимо получить *список городов, которые посещали люди, посещавшие так же город X,* в виде *массива объектов модели City*:

```
[<City>, ...]
```

Вложенные запросы

На чистом SQL:

```
City.find_by_sql("
SELECT * FROM `cities`
WHERE `cities`.`id` IN (
   SELECT `v1`.`city_id` FROM `visits` as `v1`
   WHERE `v1`.`person_id` IN(
      SELECT `v2`.`person_id` FROM `visits` as `v2`
      WHERE `city_id` = #{city_id}
   )
)
")
```

Вложенные запросы

На чистом SQL:

```
City.find_by_sql("
SELECT * FROM `cities`
WHERE `cities`.`id` IN (
   SELECT `v1`.`city_id` FROM `visits` as `v1`
   WHERE `v1`.`person_id` IN(
      SELECT `v2`.`person_id` FROM `visits` as `v2`
      WHERE `city_id` = #{city_id}
   )
)
")
```

Вложенные запросы

На чистом SQL:

```
City.find_by_sql("
SELECT * FROM `cities`
WHERE `cities`.`id` IN (
   SELECT `v1`.`city_id` FROM `visits` as `v1`
   WHERE `v1`.`person_id` IN(
      SELECT `v2`.`person_id` FROM `visits` as `v2`
      WHERE `city_id` = #{city_id}
   )
)
")
```

Вложенные запросы

На чистом SQL:

```
City.find_by_sql("
SELECT * FROM `cities`
WHERE `cities`.`id` IN (
    SELECT `v1`.`city_id` FROM `visits` as `v1`
    WHERE `v1`.`person_id` IN(
        SELECT `v2`.`person_id` FROM `visits` as `v2`
        WHERE `city_id` = #{city_id}
    )
)
")
```

Вложенные запросы

ARel:

```
at = City.arel_table

v1_alias = Visit.arel_table.alias("v1")
v1 = Arel::SelectManager.new(ActiveRecord::Base) \
                         .from(v1_alias)

v2_alias = Visit.arel_table.alias("v2")
v2 = Arel::SelectManager.new(ActiveRecord::Base) \
                         .from(v2_alias)
```

Вложенные запросы

ARel:

```
at = City.arel_table

v1_alias = Visit.arel_table.alias("v1")
v1 = Arel::SelectManager.new(ActiveRecord::Base) \
                         .from(v1_alias)

v2_alias = Visit.arel_table.alias("v2")
v2 = Arel::SelectManager.new(ActiveRecord::Base) \
                         .from(v2_alias)
```

Вложенные запросы

ARel:

```
at = City.arel_table

v1_alias = Visit.arel_table.alias("v1")
v1 = Arel::SelectManager.new(ActiveRecord::Base) \
                         .from(v1_alias)

v2_alias = Visit.arel_table.alias("v2")
v2 = Arel::SelectManager.new(ActiveRecord::Base) \
                         .from(v2_alias)
```

Вложенные запросы

ARel:

```
at = City.arel_table

v1_alias = Visit.arel_table.alias("v1")
v1 = Arel::SelectManager.new(ActiveRecord::Base) \
                         .from(v1_alias)

v2_alias = Visit.arel_table.alias("v2")
v2 = Arel::SelectManager.new(ActiveRecord::Base) \
                         .from(v2_alias)
```

Вложенные запросы

ARel:

```
City.find_by_sql(
    at.project(Arel.star).where(
        at[:id].in(
            v1.project(v1_alias[:city_id]).where(
                v1_alias[:person_id].in(
                    v2.project(v2_alias[:person_id]).where(
                        v2_alias[:city_id].eq(city_id)
                    )
                )
            )
        )
    )
)
```

Вложенные запросы

```
at.project("*")
at.project(Arel::SqlLiteral.new("*"))
# .to_sql => "SELECT * FROM `cities`"

~~at.project(at["*"])~~
at.project(at[Arel.star])
# .to_sql => "SELECT `cities`.* FROM `cities`"
```

ARel:

```
City.find_by_sql(
    at.project(Arel.star).where(
        at[:id].in(
            v1.project(v1_alias[:city_id]).where(
                v1_alias[:person_id].in(
                    v2.project(v2_alias[:person_id]).where(
                        v2_alias[:city_id].eq(city_id)
                    )
                )
            )
        )
    )
)
```

Вложенные запросы

ARel:

```
City.find_by_sql(
    at.project(Arel.star).where(
        at[:id].in(
            v1.project(v1_alias[:city_id]).where(
                v1_alias[:person_id].in(
                    v2.project(v2_alias[:person_id]).where(
                        v2_alias[:city_id].eq(city_id)
                    )
                )
            )
        )
    )
)
```

```
Arel::Nodes::Equality.new(v2_alias[:city_id], cities[1].id)
# .to_sql => "`v2`.`city_id` = X"
```

Вложенные запросы

ARel:

```
Arel::Nodes::In.new(at[:id], v1.…)
# .to_sql => "`cities`.`id` IN (SELECT …)"
```

```
City.find_by_sql(
    at.project(Arel.star).where(
        at[:id].in(
            v1.project(v1_alias[:city_id]).where(
                v1_alias[:person_id].in(
                    v2.project(v2_alias[:person_id]).where(
                        v2_alias[:city_id].eq(city_id)
                    )
                )
            )
        )
    )
)
```

SQL

Вложенные запросы

ARel:

НЕ ОЧЕНЬ

Я думал, намного будет… Намного лучше будет это все

Сколько раз сюда ходи-и-и-л — было намного лучше, но на этот раз как-то не удало-о-ось.

Вложенные запросы

ActiveRecord only:

```
City.where(
    :id => Visit.select(:city_id).where(
        :person_id => Visit.select(:person_id) \
                                .where(:city_id=>city_id)
    )
)
```

Вложенные запросы

ActiveRecord only:

```
City.where(
    :id => Visit.select(:city_id).where(
        :person_id => Visit.select(:person_id) \
                                .where(:city_id=>city_id)
    )
)
```

Вложенные запросы

ActiveRecord only:

```
City.where(
    :id => Visit.select(:city_id).where(
        :person_id => Visit.select(:person_id) \
                                .where(:city_id=>city_id)
    )
)
```

Вложенные запросы

ActiveRecord only:

SO GOOD!

Вложенные запросы с агрегацией

Достаточно часто используемый пример: статистика скачиваний файлов.

DownloadStatistic	
day_in_tz	integer
hour_in_tz	integer
ip	integer
minute_in_tz	integer
month_in_tz	integer
year_in_tz	integer
created_at	datetime
file_id	integer
id	integer
updated_at	datetime

Вложенные запросы с агрегацией

Необходимо получить *статистику уникальных загрузок файлов с группировкой по дням* в виде:

```
[
    {
     :file_id=>5,
     :year_in_tz=>2012,
     :month_in_tz=>3,
     :day_in_tz=>1,
     :hosts_count=>2
    }, …
]
```

Вложенные запросы с агрегацией

ActiveRecord + ARel

```
DownloadStatistic.select([:file_id, :year_in_tz,
                          :month_in_tz, :day_in_tz,
                          Arel.star.count.as("hosts_count")]) \
                 .from( DownloadStatistic.select(
                          [:file_id,:year_in_tz, :month_in_tz, :day_in_tz]
                        ).group(:file_id, :ip,
                                :year_in_tz,:month_in_tz, :day_in_tz) \
                         .arel.as("t1")
                 ).group(:file_id, :year_in_tz, :month_in_tz, :day_in_tz)
```

Вложенные запросы с агрегацией

ActiveRecord + ARel

```
DownloadStatistic.select([:file_id, :year_in_tz,
                          :month_in_tz, :day_in_tz,
                          Arel.star.count.as("hosts_count")]) \
                 .from( DownloadStatistic.select(
                          [:file_id,:year_in_tz, :month_in_tz, :day_in_tz]
                        ).group(:file_id, :ip,
                                :year_in_tz,:month_in_tz, :day_in_tz) \
                         .arel.as("t1")
                 ).group(:file_id, :year_in_tz, :month_in_tz, :day_in_tz)
```

Вложенные запросы с агрегацией

ActiveRecord + ARel

```
DownloadStatistic.select([:file_id, :year_in_tz,
                          :month_in_tz, :day_in_tz,
                          Arel.star.count.as("hosts_count")]) \
                 .from( DownloadStatistic.select(
                          [:file_id,:year_in_tz, :month_in_tz, :day_in_tz]
                        ).group(:file_id, :ip,
                                :year_in_tz,:month_in_tz, :day_in_tz) \
                          .arel.as("t1")
                 ).group(:file_id, :year_in_tz, :month_in_tz, :day_in_tz)
```

<Arel::SelectManager>

Вложенные запросы с агрегацией

ActiveRecord + ARel

```
DownloadStatistic.select([:file_id, :year_in_tz,
                          :month_in_tz, :day_in_tz,
                          Arel.star.count.as("hosts_count")]) \
                 .from( DownloadStatistic.select(
                          [:file_id,:year_in_tz, :month_in_tz, :day_in_tz]
                        ).group(:file_id, :ip,
                                :year_in_tz,:month_in_tz, :day_in_tz) \
                         .arel. as("t1")
                 ).group(:file_id, :year_in_tz, :month_in_tz, :day_in_tz)
```

```
<Arel::Nodes::TableAlias>
# .to_sql => "(SELECT …) t1"
```

Вложенные запросы с агрегацией

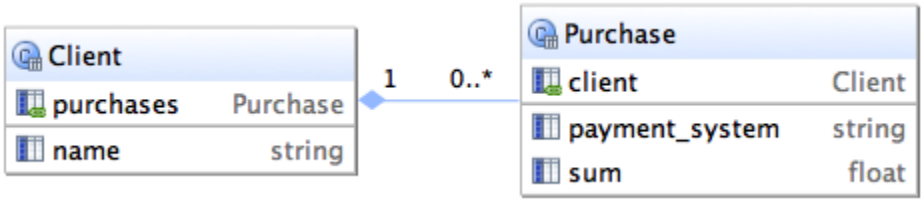
ActiveRecord + ARel

```
# .to_sql => COUNT(*) as `hosts_count`
```

```
DownloadStatistic.select([:file_id, :year_in_tz,
                          :month_in_tz, :day_in_tz,
                          Arel.star.count.as("hosts_count")]) \
                 .from( DownloadStatistic.select(
                          [:file_id,:year_in_tz, :month_in_tz, :day_in_tz]
                        ).group(:file_id, :ip,
                                :year_in_tz,:month_in_tz, :day_in_tz) \
                         .arel.as("t1")
                 ).group(:file_id, :year_in_tz, :month_in_tz, :day_in_tz)
```

JOIN-ы таблиц

Возьмём в качестве примера интернет-магазин. В данный момент нас интересуют следующие таблицы и связь между ними:



JOIN-ы таблиц

Необходимо получить массив объектов моделей Client с дополнительными полями, отражающими общее количество покупок и сумму потраченных в магазине средств в виде:

```
[
  <Client … spent_money=2000
            purchases_count=3>,
  …
]
```

JOIN-ы таблиц: add_select

Иногда необходимо динамически добавлять к запросу поля выборки:

```
class ActiveRecord::Relation
  def add_select(selects)
    t=self.scoped
    selects=[selects] unless selects.kind_of?(Array)
    t.select_values = [self.arel_table[Arel.star]] if
t.select_values.blank?
    t.select_values+=selects
    t
  end
end
```

JOIN-ы таблиц: add_select

Иногда необходимо динамически добавлять к запросу поля выборки:

```
class ActiveRecord::Relation
  def add_select(selects)
    t=self.scoped
    selects=[selects] unless selects.kind_of?(Array)
    t.select_values = [self.arel_table[Arel.star]] if
t.select_values.blank?
    t.select_values+=selects
    t
  end
end
```

JOIN-ы таблиц: add_select

Примеры использования:

```
p = Purchase.select(Purchase.arel_table[:sum])
# .to_sql => SELECT `purchases`.`sum` FROM `purchases`
```

```
x = p.add_select(Purchase.arel_table[:client_id])
# . to_sql => SELECT `purchases`.`sum`, `purchases`.
`client_id` FROM `purchases`
```

JOIN-ы таблиц: add_select

Примеры использования:

```
p = Purchase.select(Purchase.arel_table[:sum])
# .to_sql => SELECT `purchases`.`sum` FROM `purchases`


x = p.add_select(Purchase.arel_table[:client_id])
# . to_sql => SELECT `purchases`.`sum`, `purchases`.
`client_id` FROM `purchases`
```

JOIN-ы таблиц: add_select

Примеры использования:

```
p = Purchase.select(Purchase.arel_table[:sum])
# .to_sql => SELECT `purchases`.`sum` FROM `purchases`
```

```
x = p.add_select(Purchase.arel_table[:client_id])
# . to_sql => SELECT `purchases`.`sum`, `purchases`.
`client_id` FROM `purchases`
```

JOIN-ы таблиц: LEFT OUTER JOIN

```
p_at = Purchase.arel_table
c_at = Client.arel_table

purchases_join = Arel::OuterJoin.new(
   p_at,
   Arel::Nodes::On.new(
     c_at[:id].eq(p_at[:client_id])
   )
)
#.to_sql => "LEFT OUTER JOIN `purchases` ON `clients`.`id`
= `purchases`.`client_id`"
```

JOIN-ы таблиц: LEFT OUTER JOIN

```
p_at = Purchase.arel_table
c_at = Client.arel_table

purchases_join = Arel::OuterJoin.new(
   p_at,
   Arel::Nodes::On.new(
     c_at[:id].eq(p_at[:client_id])
   )
)
#.to_sql => "LEFT OUTER JOIN `purchases` ON `clients`.`id`
= `purchases`.`client_id`"
```

JOIN-ы таблиц: LEFT OUTER JOIN

```
p_at = Purchase.arel_table
c_at = Client.arel_table

purchases_join = Arel::OuterJoin.new(
   p_at,
   Arel::Nodes::On.new(
     c_at[:id].eq(p_at[:client_id])
   )
)
#.to_sql => "LEFT OUTER JOIN `purchases` ON `clients`.`id`
= `purchases`.`client_id`"
```

JOIN-ы таблиц: LEFT OUTER JOIN

```
p_at = Purchase.arel_table
c_at = Client.arel_table

purchases_join = Arel::OuterJoin.new(
   p_at,
   Arel::Nodes::On.new(
     c_at[:id].eq(p_at[:client_id])
   )
)
#.to_sql => "LEFT OUTER JOIN `purchases` ON `clients`.`id`
= `purchases`.`client_id`"
```

JOIN-ы таблиц: Полные запросы

```
#p_at, c_at, purchases_join определили ранее

Client.joins(purchases_join) \
      .add_select(
        p_at[:id].count \
        .as("purchases_count")
      ).group(p_at[:client_id])

# .to_sql => "SELECT `clients`.*, COUNT(`purchases`.`id`)
AS purchases_count FROM `clients` LEFT OUTER JOIN
`purchases` ON `clients`.`id` = `purchases`.`client_id`
GROUP BY `purchases`.`client_id`"
```

JOIN-ы таблиц: Полные запросы

```
#p_at, c_at, purchases_join определили ранее

Client.joins(purchases_join) \
      .add_select(
        p_at[:id].count \
        .as("purchases_count")
      ).group(p_at[:client_id])

# .to_sql => "SELECT `clients`.*, COUNT(`purchases`.`id`)
AS purchases_count FROM `clients` LEFT OUTER JOIN
`purchases` ON `clients`.`id` = `purchases`.`client_id`
GROUP BY `purchases`.`client_id`"
```

JOIN-ы таблиц: Полные запросы

```
#p_at, c_at, purchases_join определили ранее

Client.joins(purchases_join) \
      .add_select(
        p_at[:id].count \
        .as("purchases_count")
      ).group(p_at[:client_id])

# .to_sql => "SELECT `clients`.*, COUNT(`purchases`.`id`)
AS purchases_count FROM `clients` LEFT OUTER JOIN
`purchases` ON `clients`.`id` = `purchases`.`client_id`
GROUP BY `purchases`.`client_id`"
```

JOIN-ы таблиц: Полные запросы

```
#p_at, c_at, purchases_join определили ранее

Client.joins(purchases_join) \
      .add_select(
        p_at[:id].count \
        .as("purchases_count")
      ).group(p_at[:client_id])

# .to_sql => "SELECT `clients`.*, COUNT(`purchases`.`id`)
AS purchases_count FROM `clients` LEFT OUTER JOIN
`purchases` ON `clients`.`id` = `purchases`.`client_id`
GROUP BY `purchases`.`client_id`"
```

JOIN-ы таблиц: Полные запросы

Аналогично для spent_money:

```
Client.joins(purchases_join) \
      .add_select(
        p_at[:sum].sum.as("spent_money")
      ).group(p_at[:client_id])
```

```
# .to_sql => "SELECT `clients`.*, SUM(`purchases`.`sum`) AS
spent_money FROM `clients` LEFT OUTER JOIN `purchases` ON
`clients`.`id` = `purchases`.`client_id` GROUP BY
`purchases`.`client_id`"
```

JOIN-ы таблиц: Полные запросы

Аналогично для spent_money:

```
Client.joins(purchases_join) \
      .add_select(
        p_at[:sum].sum.as("spent_money")
      ).group(p_at[:client_id])
```

```
# .to_sql => "SELECT `clients`.*, SUM(`purchases`.`sum`) AS
spent_money FROM `clients` LEFT OUTER JOIN `purchases` ON
`clients`.`id` = `purchases`.`client_id` GROUP BY
`purchases`.`client_id`"
```

JOIN-ы таблиц: Scopes

joins_purchase в виде *named scope* с предотвращением повторных join-ов:

```
class Client < ActiveRecord::Base
…
  scope :joins_purchase, lambda{
    unless defined?(@joins_purchase)
      p = Purchase.arel_table
      @joins_purchase = joins(Arel::OuterJoin.new(p, Arel::
Nodes::On.new(arel_table[:id].eq(p[:client_id])))).group(p
[:client_id])
    end
    @joins_purchase
  }
…
```

JOIN-ы таблиц: Scopes

```
class Client < ActiveRecord::Base
…
  scope :with_purchases_count, lambda{
    joins_purchase.add_select(
      Purchase.arel_table[:id].count.as("purchases_count")
    )
  }

  scope :with_spent_money, lambda{
    joins_purchase.add_select(
      Purchase.arel_table[:sum].sum.as("spent_money")
    )
  }
…
```

JOIN-ы таблиц: Scopes

```
class Client < ActiveRecord::Base
…
  scope :with_purchases_count, lambda{
    joins_purchase.add_select(
      Purchase.arel_table[:id].count.as("purchases_count")
    )
  }

  scope :with_spent_money, lambda{
    joins_purchase.add_select(
      Purchase.arel_table[:sum].sum.as("spent_money")
    )
  }
…
```

JOIN-ы таблиц: Scopes

```
class Client < ActiveRecord::Base
…
  scope :with_purchases_count, lambda{
    joins_purchase.add_select(
      Purchase.arel_table[:id].count.as("purchases_count")
    )
  }

  scope :with_spent_money, lambda{
    joins_purchase.add_select(
      Purchase.arel_table[:sum].sum.as("spent_money")
    )
  }
…
```

JOIN-ы таблиц: Scopes

```
class Client < ActiveRecord::Base
…
  scope :with_purchases_information, lambda{
    with_purchases_count.with_spent_money
  }
…
end
Client.with_purchases_information
# .to_sql => "
#SELECT `clients`.*, COUNT(`purchases`.`id`) AS purchases_count,
#        SUM(`purchases`.`sum`) AS spent_money
#FROM `clients`
#LEFT OUTER JOIN `purchases` ON `clients`.`id` = `purchases`.`client_id`
#GROUP BY `purchases`.`client_id`
#"
```

JOIN-ы таблиц: Проблема

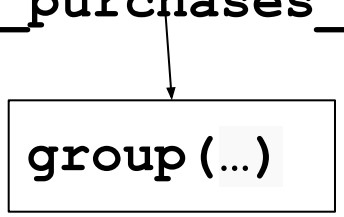
```
Client.with_purchases_information.count
```

group(…)

```
# => { group_item=>count, … }
```

JOIN-ы таблиц: Проблема

```
Client.with_purchases_information.count
```



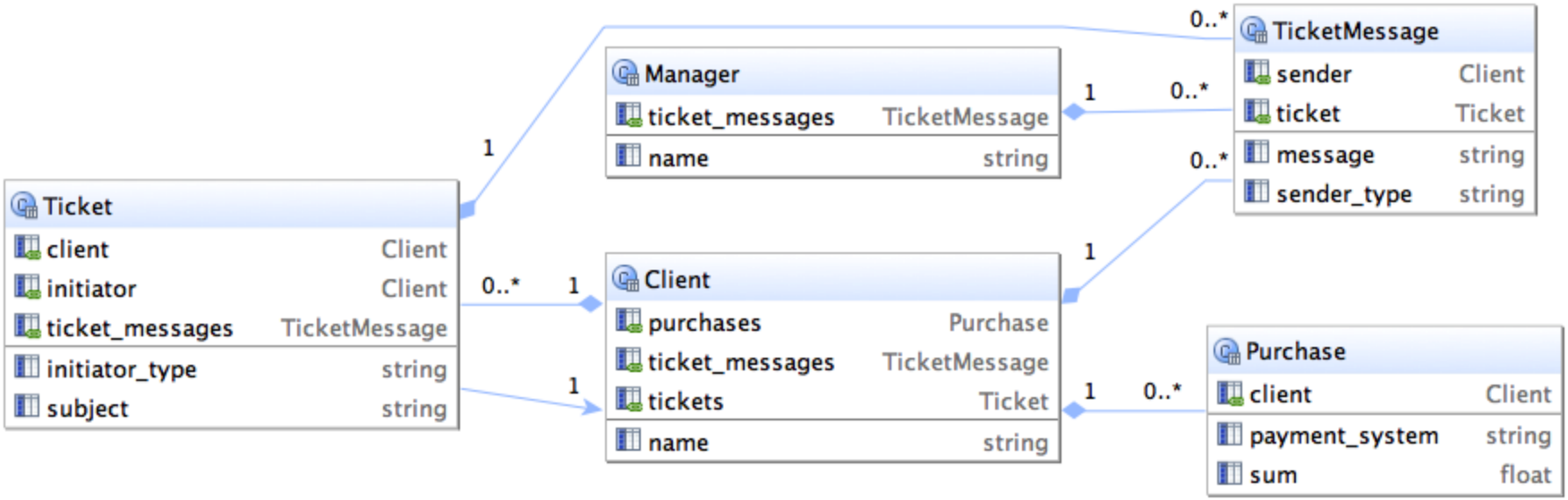

```
# => { group_item=>count, … }
```

Решения:

- Переписать со вложенными запросами
- Считать количество не COUNT(*) запросом, а на стороне Rails: `Client.with_purchases_information.all.count`

Polymorphic Associations

Используем тот же самый магазин, добавим к нему систему тикетов (общение менеджеров с клиентами)



Polymorphic Associations

```
class Ticket < ActiveRecord::Base
  belongs_to :client
  belongs_to :initiator, :polymorphic => true
  has_many :ticket_messages
end

class TicketMessage < ActiveRecord::Base
  belongs_to :sender, :polymorphic => true
  belongs_to :ticket
end

class Manager < ActiveRecord::Base
  has_many :ticket_messages, :as=>:sender
end
```

Polymorphic Associations

Необходимо получить *массив сообщений для данного тикета* с указанием *имени отправителя* в виде:

```
[
  <TicketMessage … sender_name="Василий">,
  …
]
```

Polymorphic Associations: JOINS

```
c_at = Client.arel_table
tm_at = TicketMessage.arel_table


sender_client_join = Arel::Nodes::OuterJoin.new(
  c_at,
  Arel::Nodes::On.new(
    c_at[:id].eq(tm_at[:sender_id]). \
    and(tm_at[:sender_type].eq("Client"))
  )
)
```

Polymorphic Associations: JOINS

```
c_at = Client.arel_table
tm_at = TicketMessage.arel_table


sender_client_join = Arel::Nodes::OuterJoin.new(
  c_at,
  Arel::Nodes::On.new(
    c_at[:id].eq(tm_at[:sender_id]). \
    and(tm_at[:sender_type].eq("Client"))
  )
)
```

Polymorphic Associations: JOINS

```
m_at = Manager.arel_table
tm_at = TicketMessage.arel_table


sender_manager_join = Arel::Nodes::OuterJoin.new(
  m_at,
  Arel::Nodes::On.new(
    m_at[:id].eq(tm_at[:sender_id]). \
    and(tm_at[:sender_type].eq("Manager"))
  )
)
```

Polymorphic Associations: JOINS

```
m_at = Manager.arel_table
tm_at = TicketMessage.arel_table


sender_manager_join = Arel::Nodes::OuterJoin.new(
  m_at,
  Arel::Nodes::On.new(
    m_at[:id].eq(tm_at[:sender_id]). \
    and(tm_at[:sender_type].eq("Manager"))
  )
)
```

Polymorphic Associations: Запрос

Итоговый запрос:

```
ticket = Ticket.find(some_id)
ticket.ticket_messages \
      .joins(sender_client_join).joins(sender_manager_join)
\
      .add_select(
        Arel::Nodes::NamedFunction.new("IFNULL",
        [
          c_at[:name],
          m_at[:name]
        ]).as("sender_name")
      )
```

Polymorphic Associations: Запрос

Итоговый запрос:

```
ticket = Ticket.find(some_id)
ticket.ticket_messages \
      .joins(sender_client_join).joins(sender_manager_join)
\
      .add_select(
        Arel::Nodes::NamedFunction.new("IFNULL",
        [
          c_at[:name],
          m_at[:name]
        ]).as("sender_name")
      )
```

```
# .to_sql => "IFNULL(`clients`.`name`, `managers`.`name`) AS sender_name"
```

Ну и хватит с запросами

Полезные ruby gems

- **Squeel** - sql запросы в ruby стиле `https://github.com/ernie/squeel`
- **Advanjo** - сложные join-ы в одну строку `https://github.com/rap-kasta/advanjo.git`
- **Chooseme** - подбор индекса по используемым столбцам `https://github.com/rap-kasta/chooseme.git`

ВОПРОСЫ?

Контакты

Манылов Павел Сергеевич

email: rapkasta@gmail.com
jabber: rapkasta@jabber.ru
habrahabr: rapkasta
twitter: rap_kasta

Нет, я не фанат, ну ... не на столько.