Analytics en tiempo real con Rails

Eduardo Hernández

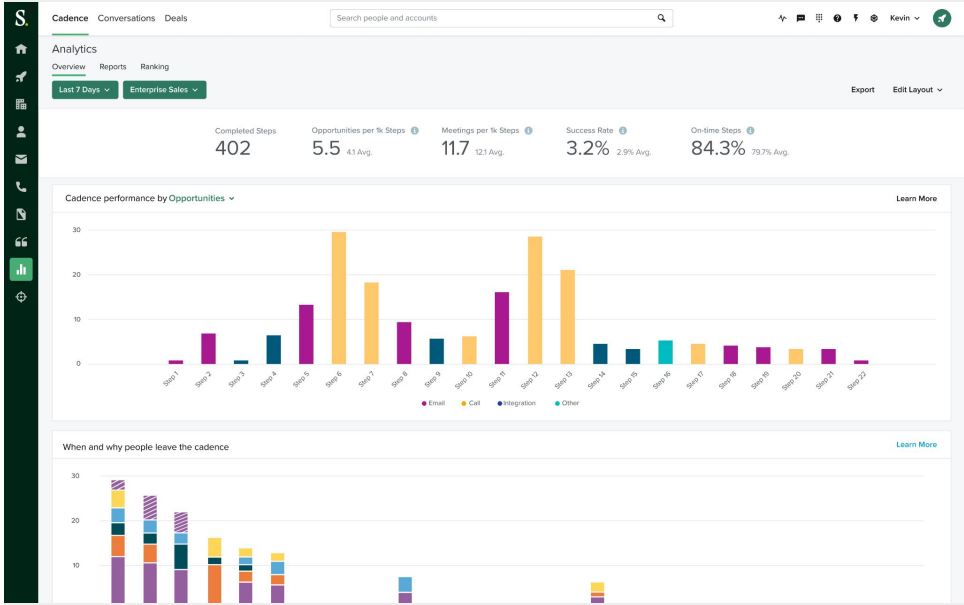
Software Engineer @ Salesloft

Co-founder de Calzada Code

Colaborador en Codeando México

Analytics APIs

Interfaz qué facilite el acceso a información resumida



REST para extraer métricas?

GET /orders?<query>

GET /orders/:id

POST /orders

PATCH /orders/:id

DELETE /orders/:id

¿Cuánto fue el total de las ventas (brutas) esta semana?

¿Cuánto fue el total de ventas por sucursal?

¿Cuánto fue el total de impuestos a pagar este mes?

REST para extraer métricas?

GET /orders?<query>

GET /orders/:id

POST /orders

PATCH /orders/:id

DELETE /orders/:id

¿Cuánto fue el total de ventas (brutas) esta semana?

¿Cuánto fue el total de ventas por sucursal?

¿Cuánto fue el total de cargos por envío de este mes?

Metrics:

- Ventas Brutas - Impuestos

Dimensions:

- Sucursales - Fecha

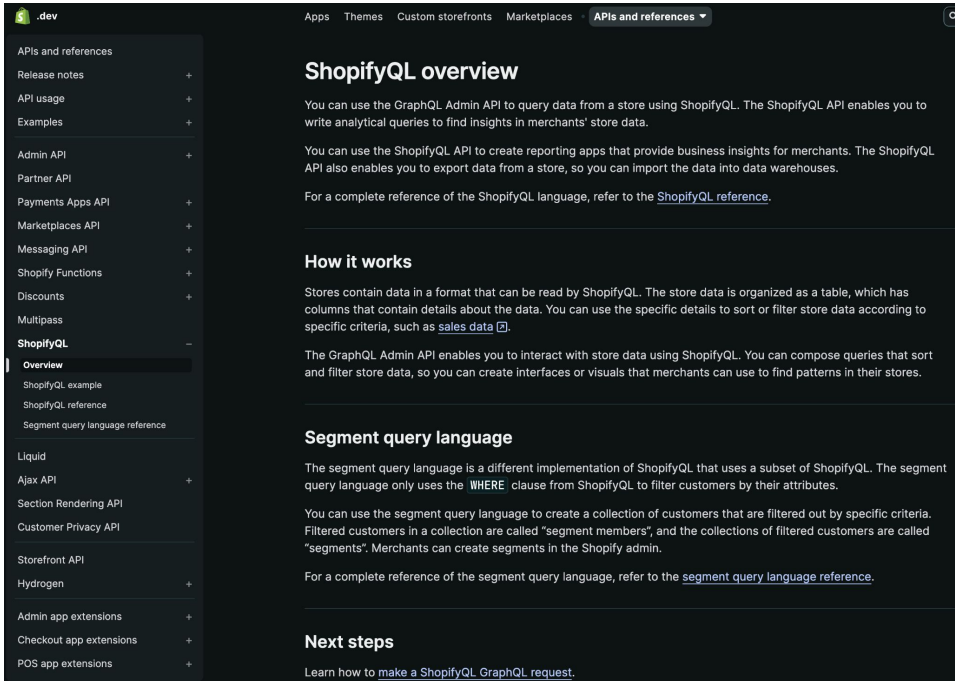
Metrics & Dimensions

```sql
-- Metrics: Gross Sales, Shipping Rate
-- Dimensions: Date, Branch

SELECT sum(amount), sum(shipping_rate),
FROM ordes
WHERE completed_at > beggining_of_week AND completed_at < end_of_week
GROUP BY date, branch;
```

ShopifyQL

> The ShopifyQL API enables you to write analytical queries to find insights in merchants' store data.



Aggregates are **pre-defined aggregates of numeric values**, to replicate metrics as available throughout Shopify.

Aggregates can be grouped or filtered by any of the **dimensional attributes.**

ShopifyQL

> The ShopifyQL API enables you to write analytical queries to find insights in merchants' store data.

```
FROM {table_name}

SHOW {column1, column2, ...}

  AS {alias1, alias2, ...}

VISUALIZE {column1, column2, ...}

  TYPE {visualization_type}

BY {dimension|date}

OVER {date}

WHERE {condition}

SINCE {date_offset}

UNTIL {date_offset}

ORDER BY {column1 ASC|DESC, column2 ASC|DESC, ...}

LIMIT {number}
```

ShopifyQL

> The ShopifyQL API enables you to write analytical queries to find insights in merchants' store data.

```
FROM sales
SHOW gross_sales
BY branch
SINCE '2022-06-20'
UNTIL '2022-06-25'
```

```
FROM {table_name}

SHOW {column1, column2, ...}

  AS {alias1, alias2, ...}

VISUALIZE {column1, column2, ...}

  TYPE {visualization_type}

BY {dimension|date}

OVER {date}

WHERE {condition}

SINCE {date_offset}

UNTIL {date_offset}

ORDER BY {column1 ASC|DESC, column2 ASC|DESC, ...}

LIMIT {number}
```

POST /report

```
{
  "query": {
    "show": ["<aggregates>"],
    "from": ["<source>"],
    "by": ["<dimensions>"],
    "where": {"<dimesion>": "<value>"},
    "order": [{"<column>": "asc"}],
    "since": "<date>",
    "until": "<date>"
  }
}
```

cube.js

Cube is the headless business intelligence platform. It helps data engineers and application developers access data from modern data stores, organize it into consistent definitions, and deliver it to every application.

cube examples | Real Time Dashboard

Discuss on Slack | Get started with Cube

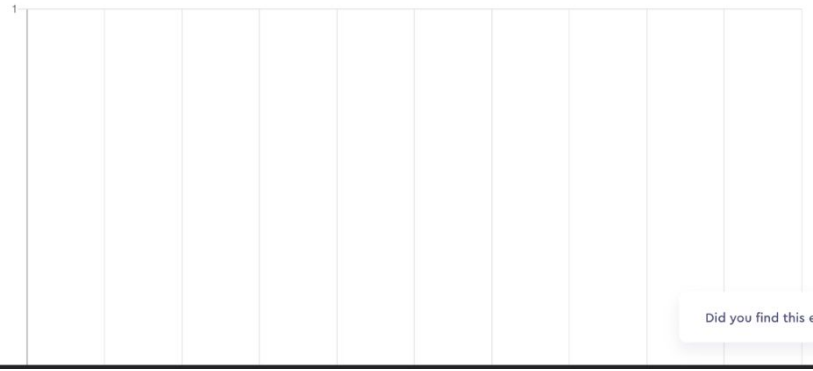
Real-Time Dashboard

This live demo shows a real-time data fetch built with Cube, MongoDB, and React.
You can use it to update charts as new data comes in.
Follow the guide or explore the source code to learn more.

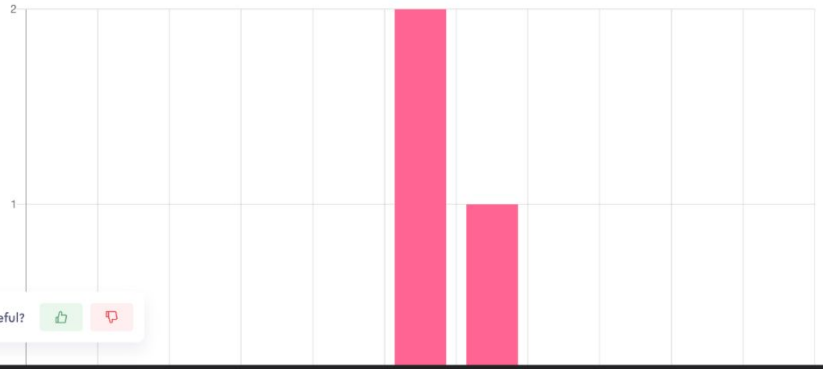
Send Button Click Event

Users Online	Total Button Clicks	Total Page Views
0	0	7

Real Time Events



Events per Minute ago



Did you find this example useful?

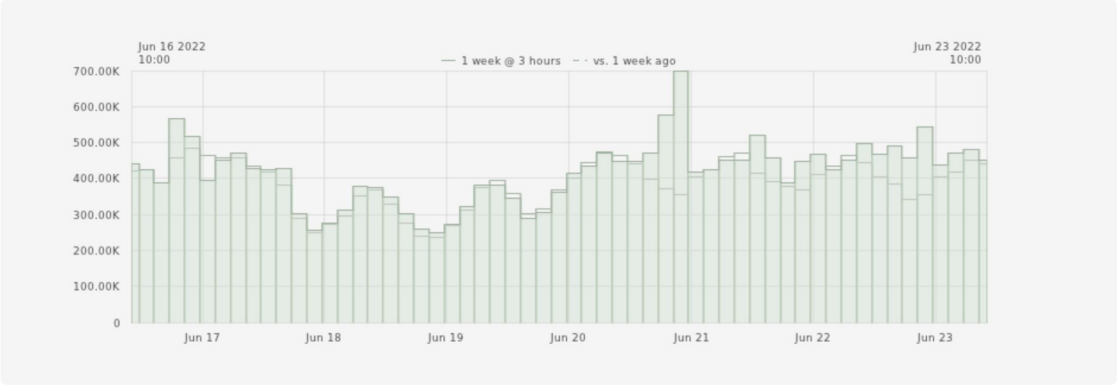
Real-time analytics with PostgreSQL by Marco Slot

Real-time analytics with PostgreSQL

- Direct Query to Raw Data
- Materialized Views
- Cache / Cascading aggregates

GH Archive

Star Tweet



Open-source developers all over the world are working on millions of projects: writing code & documentation, fixing & submitting bugs, and so forth. GH Archive is a project to **record** the public GitHub timeline, **archive it**, and **make it easily accessible** for further analysis.

GitHub provides 20+ event types, which range from new commits and fork events, to opening new tickets, commenting, and adding members to a project. These events are aggregated into hourly archives, which you can access with any HTTP client:

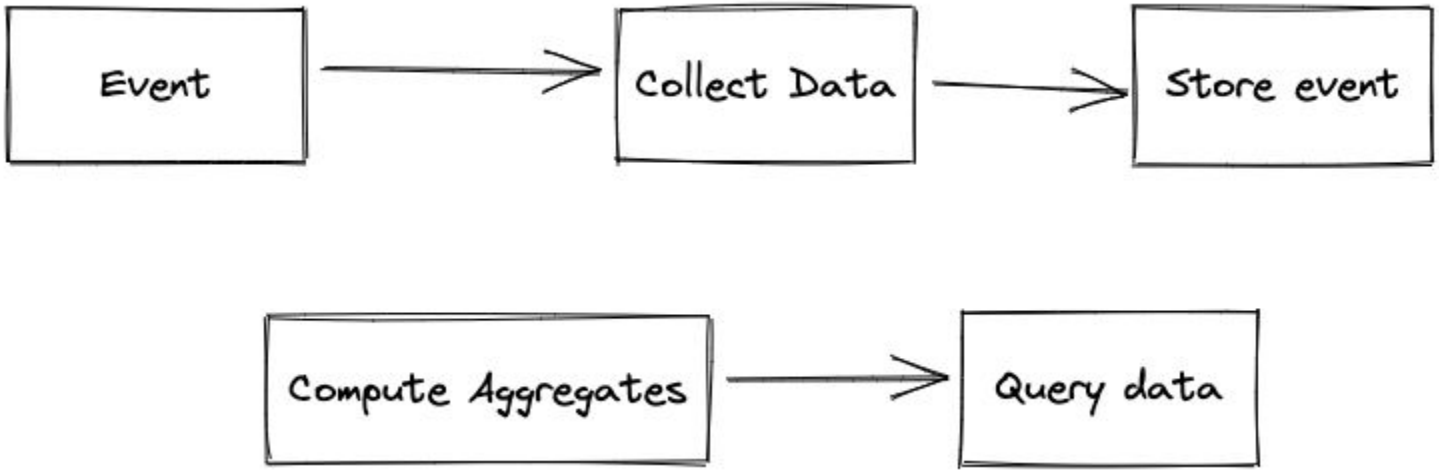
Query	Command
Activity for 1/1/2015 @ 3PM UTC	`wget https://data.gharchive.org/2015-01-01-15.json.gz`
Activity for 1/1/2015	`wget https://data.gharchive.org/2015-01-01-{0..23}.json.gz`
Activity for all of January 2015	`wget https://data.gharchive.org/2015-01-{01..31}-{0..23}.json.gz`

GitHub::Event

- Raw
- Materialized
- Cascading Aggregates


```
{
    "id": "19543559179",
    "type": "PushEvent",
    "actor": {
        "id": 6274807,
        "login": "wazho",
        "display_login": "wazho",
        "gravatar_id": "",
        "url": "https://api.github.com/users/wazho",
        "avatar_url": "https://avatars.githubusercontent.com/u/6274807?"
    },
    "repo": {
        "id": 324390811,
        "name": "pmgo-professor-willow/data-youtuber",
        "url": "https://api.github.com/repos/pmgo-professor-willow/data-youtuber"
    },
    "payload": {
        "push_id": 8733866699,
        "size": 1,
        "distinct_size": 1,
        "ref": "refs/heads/gh-pages",
        "head": "0f3c0f757ad0d42ff2a6a197c095d6d1bdf1e4af",
        "before": "1f7178fbf153b1ff5a22f2f827dd54227925094b",
        "commits": [
            {
                "sha": "0f3c0f757ad0d42ff2a6a197c095d6d1bdf1e4af",
                "author": {
                    "email": "githubaction@fake.com",
                    "name": "githubaction"
                },
                "message": "deploy to GitHub pages",
                "distinct": true,
                "url": "https://api.github.com/repos/pmgo-professor-willow/data-
youtuber/commits/0f3c0f757ad0d42ff2a6a197c095d6d1bdf1e4af"
            }
        ]
    },
    "public": true,
    "created_at": "2022-01-01T12:00:00Z",
    "org": {
        "id": 77191076,
        "login": "pmgo-professor-willow",
        "gravatar_id": "",
        "url": "https://api.github.com/orgs/pmgo-professor-willow",
        "avatar_url": "https://avatars.githubusercontent.com/u/77191076?"
    }
}
```

Data flow


Event
Collect Data
Store event
Compute Aggregates
Query data

Segment Data Quality

Bronze

Raw Ingestion

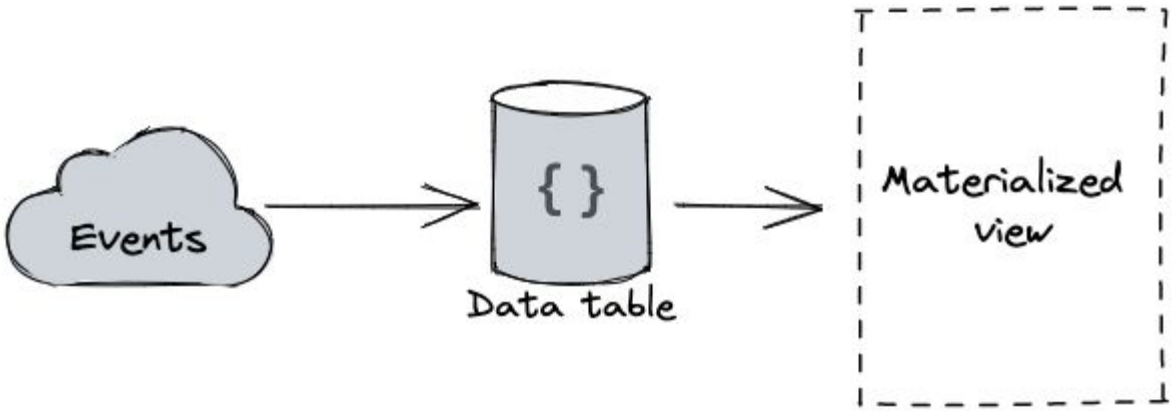
Silver

Filtered, cleaned, Augmented

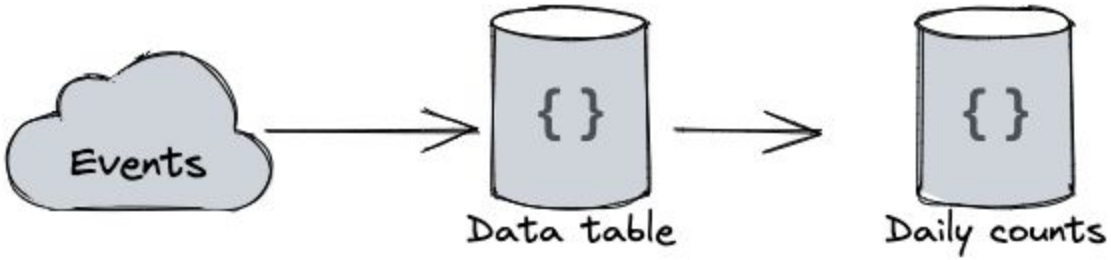
Gold

Business-level Aggregates

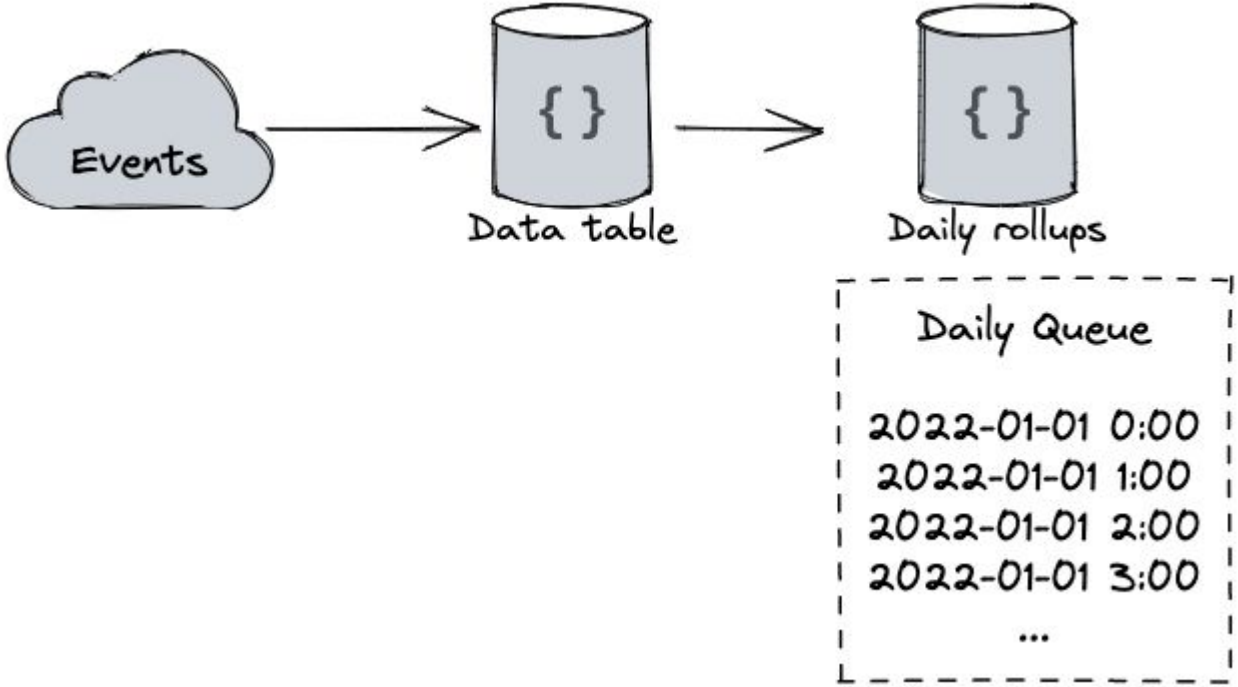
Daily counts | Materialized views



Daily counts | Actual DB table

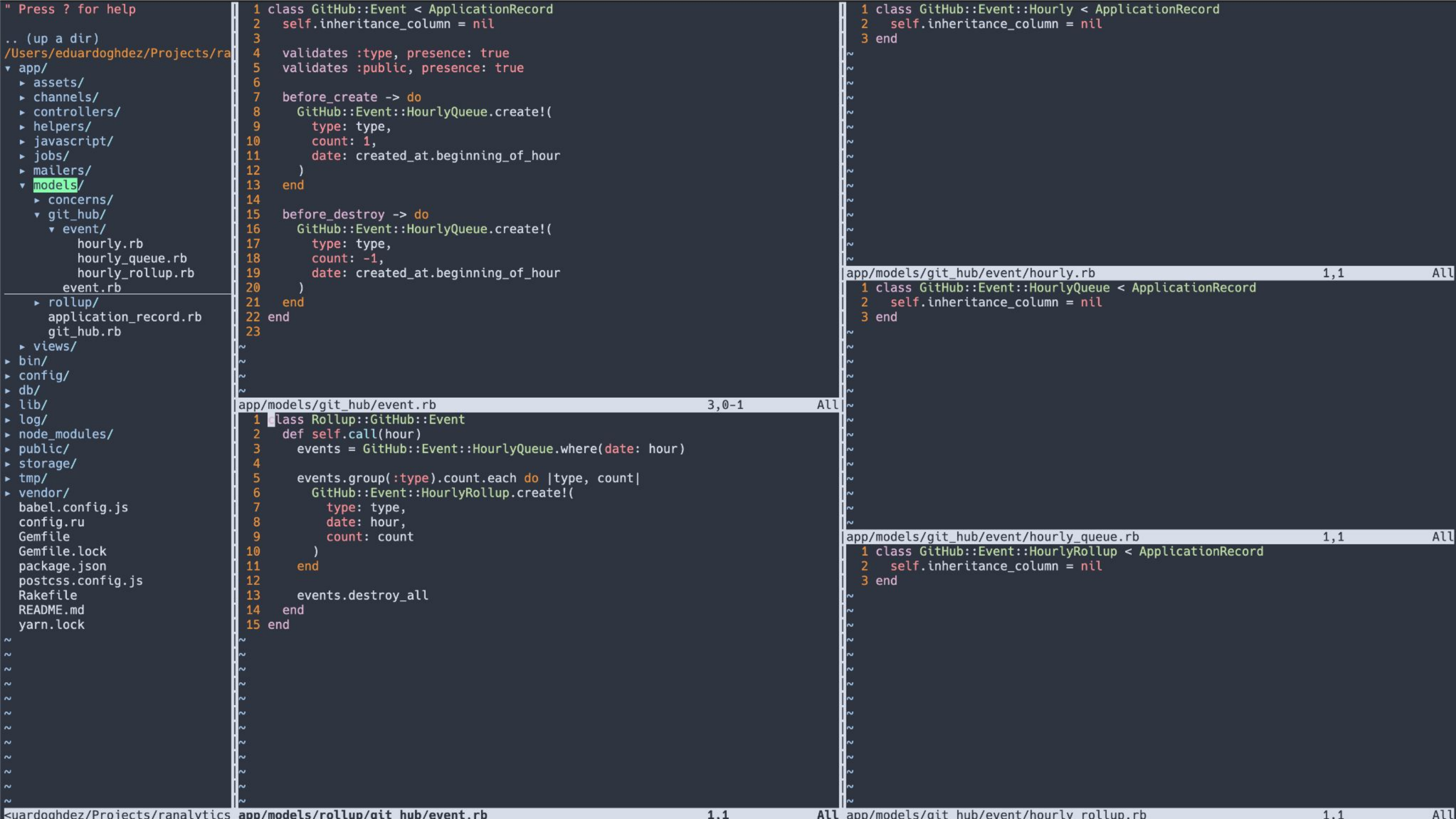


Cascading aggregates



Cascading aggregates

```
#   GitHub::Event.create(type: 'foo', created_at: '2022-01-01 1:05')
#   GitHub::Event.create(type: 'foo', created_at: '2022-01-01 1:10')
#   GitHub::Event.create(type: 'bar', created_at: '2022-01-01 2:25')
#
#   Queues the following changes in git_hub_events_queues
#
#   type    |        date         |  count
#   --------+---------------------+---------+
#     'foo' | 2022-01-01 01:00:00 |       1 |
#     'foo' | 2022-01-01 01:00:00 |       1 |
#     'bar' | 2022-01-01 02:00:00 |       1 |
#
#   Rollup the following changes in git_hub_events_rollups
#
#   type    |        date         |  count
#   --------+---------------------+---------+
#     'foo' | 2022-01-01 01:00:00 |       2 |
#     'bar' | 2022-01-01 02:00:00 |       1 |
```



```
" Press ? for help

.. (up a dir)
/Users/eduardoghdez/Projects/ra
▾ app/
  ▸ assets/
  ▸ channels/
  ▸ controllers/
  ▸ helpers/
  ▸ javascript/
  ▸ jobs/
  ▸ mailers/
  ▾ models/
    ▸ concerns/
    ▾ git_hub/
      ▾ event/
          hourly.rb
          hourly_queue.rb
          hourly_rollup.rb
        event.rb
    ▸ rollup/
      application_record.rb
      git_hub.rb
  ▸ views/
▸ bin/
▸ config/
▸ db/
▸ lib/
▸ log/
▸ node_modules/
▸ public/
▸ storage/
▸ tmp/
▸ vendor/
  babel.config.js
  config.ru
  Gemfile
  Gemfile.lock
  package.json
  postcss.config.js
  Rakefile
  README.md
  yarn.lock
<uardoghdez/Projects/ranalytics
```

```ruby
class GitHub::Event < ApplicationRecord
  self.inheritance_column = nil

  validates :type, presence: true
  validates :public, presence: true

  before_create -> do
    GitHub::Event::HourlyQueue.create!(
      type: type,
      count: 1,
      date: created_at.beginning_of_hour
    )
  end

  before_destroy -> do
    GitHub::Event::HourlyQueue.create!(
      type: type,
      count: -1,
      date: created_at.beginning_of_hour
    )
  end
end

app/models/git_hub/event.rb                                  3,0-1          All
```

```ruby
class Rollup::GitHub::Event
  def self.call(hour)
    events = GitHub::Event::HourlyQueue.where(date: hour)

    events.group(:type).count.each do |type, count|
      GitHub::Event::HourlyRollup.create!(
        type: type,
        date: hour,
        count: count
      )
    end

    events.destroy_all
  end
end

app/models/rollup/git_hub/event.rb                           1,1            All
```

```ruby
class GitHub::Event::Hourly < ApplicationRecord
  self.inheritance_column = nil
end

app/models/git_hub/event/hourly.rb                           1,1            All
```

```ruby
class GitHub::Event::HourlyQueue < ApplicationRecord
  self.inheritance_column = nil
end

app/models/git_hub/event/hourly_queue.rb                     1,1            All
```

```ruby
class GitHub::Event::HourlyRollup < ApplicationRecord
  self.inheritance_column = nil
end

app/models/git_hub/event/hourly_rollup.rb                    1,1            All
```

Schedule Hourly rollups

- Clockwork
- Whenever

```ruby
require 'yajl'

hours = 6
progress = ProgressBar.create(
  title: "2022-01-01 at 00:00 GitHub Events Import",
  total: hours
)

hours.times do |hour|
  gz = open("http://data.gharchive.org/2022-01-01-#{hour}.json.gz")
  js = Zlib::GzipReader.new(gz).read

  Yajl::Parser.new.parse(js) do |event|
    GitHub::Event.create(
      event_id: event.fetch('id'),
      type: event.fetch('type'),
      public: event.fetch('public'),
      created_at: event.fetch('created_at')
    )
  end

  time = Time.parse("2022-01-01 #{hour}:00 +0000")
  Rollup::GitHub::Event.call(time)

  progress.increment
end
```

GitHub::Event::Hourly

Gem: Scenic

> Scenic adds methods to ActiveRecord::Migration to create and manage database views in Rails.

Repo: https://github.com/scenic-views/scenic

- Versiones views
- Views in SQL
- Maps to a model
- db/views

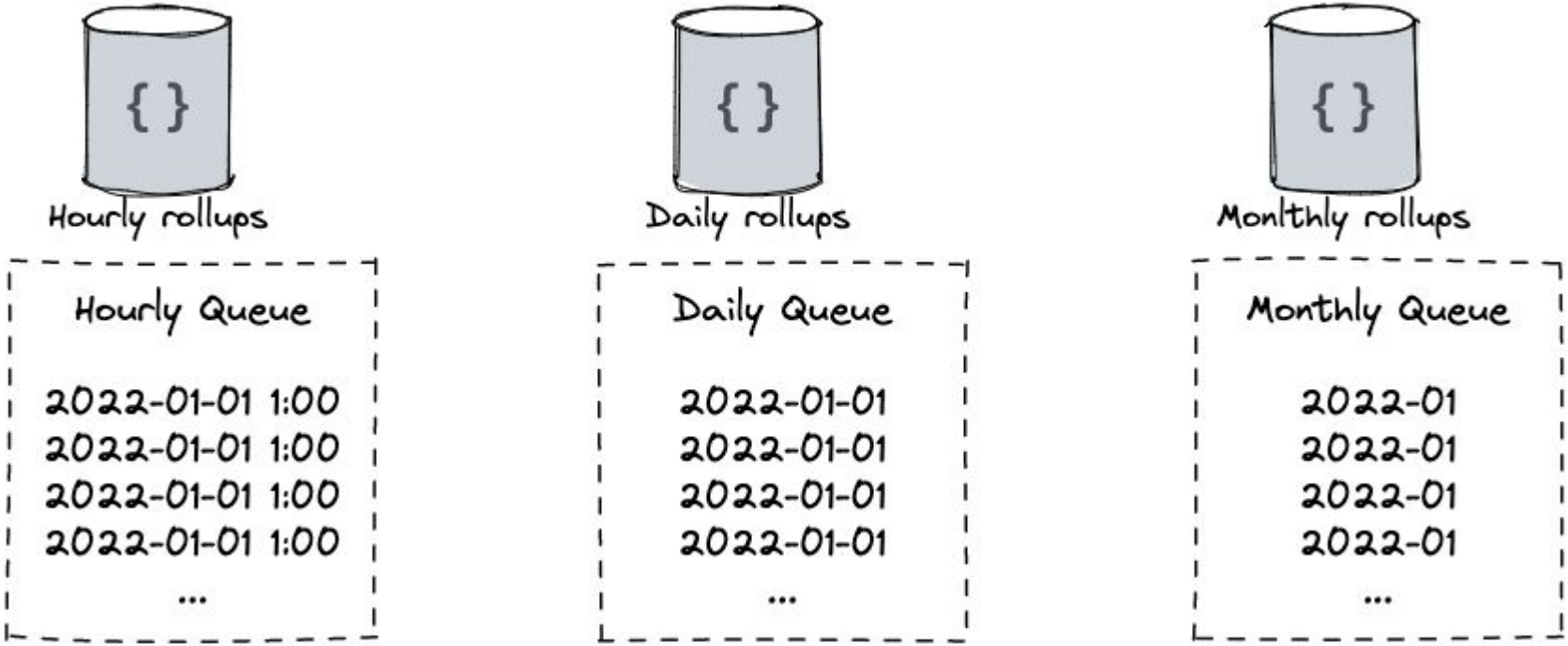
```ruby
class GitHub::Event::Hourly < ApplicationRecord
  # ...
end
```

```sql
select
  type,
  date,
  sum(count)
from (
  select
    type,
    date,
    count
  from git_hub_event_hourly_rollups

  union all

  select
    type,
    date,
    count
  from git_hub_event_hourly_queues
) git_hub_event_hourly
group by
  type,
  date
```

Cascading Aggregates



Conclusiones

- Battle tested (reponses ~36ms-300ms)
- ¿Es esto realmente real-time?
- Soluciones qué se adapten a tus necesidades
- Mantener a largo plazo este diseño se vuelve manejable con Ruby/Rails
- Versionamiento de código
- Integración continua de la aplicación
- Testeable

Gracias!

Preguntas? 😁