Configuration using the Entity Attribute Value Design Pattern

Configuration scoped by environment and entity types.

Pitch

Contributors

@erichosick

@JosueMontano10

Pitch

MagicBell? MagicBell? MagicBell?

Pitch

The challenge

- We store configuration for many integrations/channel providers.
- The configuration has many attributes of different types (integers, strings, hashes, booleans)
  - Some values are encrypted.
- Stored at a system level, per workspace, and per project (low - high precedence).
- Scoped by environment: development, staging, production, default (all environments), custom environments.

Pitch

Example

System-wide configuration

```
{
  :sender_name => "MagicBell Notifications",
  :sender_email => "noreply@magicbell-notifications.com",
  :api_key => "[FILTERED]",
  :website_push_id => "web.com.magicbell-notifications"
}
```

Workspace configuration

```
{
  :sender_name => "ACME Inc.",
  :sender_email => "noreply@example.com",
  :api_key => "[FILTERED]"
}
```

Project configuration

```
{
  :api_key => "[FILTERED]"
}
```

Pitch

Design Objectives

- Store data once
  - Example: api_key of a workspace stored for each project in the workspace.
- Generic lookup/merge functions that work for any provider (including future ones).
- Database changes only when your domain changes, not when adding new providers.
- One source of truth - everything stored in Postgres.
- High performance (sub 5 msec lookup).
- Clean DSL for saving/fetching configuration.

Pitch

Storing the configuration

```
ConfigType.create(key_name: 'sender_email', primitive_type: 'string')
ConfigType.create(key_name: 'sender_name', primitive_type: 'string')
ConfigType.create(key_name: 'api_key', encrypt: true, primitive_type: 'string')
```

```
ChannelProviderConfigBuilder.save do
  config_type_key 'api_key'
  channel_provider_slug 'sendgrid'
  value 'XXXXXXXXXXXX'
end

ChannelProviderConfigBuilder.save do
  config_type_key 'sender_name'
  channel_provider_slug 'sendgrid'
  value 'MagicBell Notifications'
end
```

Pitch

Fetching the configuration

```
ConfigAdaptor.hashify do
  channel_provider_slug 'sendgrid'
  project_id 1
end
```

Fetches values for all attributes defined for 'sendgrid' channel provider & returns a hash

Pitch

The End Result

Configuration

```
{
  :sender_name => "ACME Inc.",
  :sender_email => "noreply@example.com",
  :api_key => "[FILTERED]",
  :website_push_id => "web.com.magicbell-notifications"
}
```

The final merged configuration object - each attribute could be coming from a different level of configuration

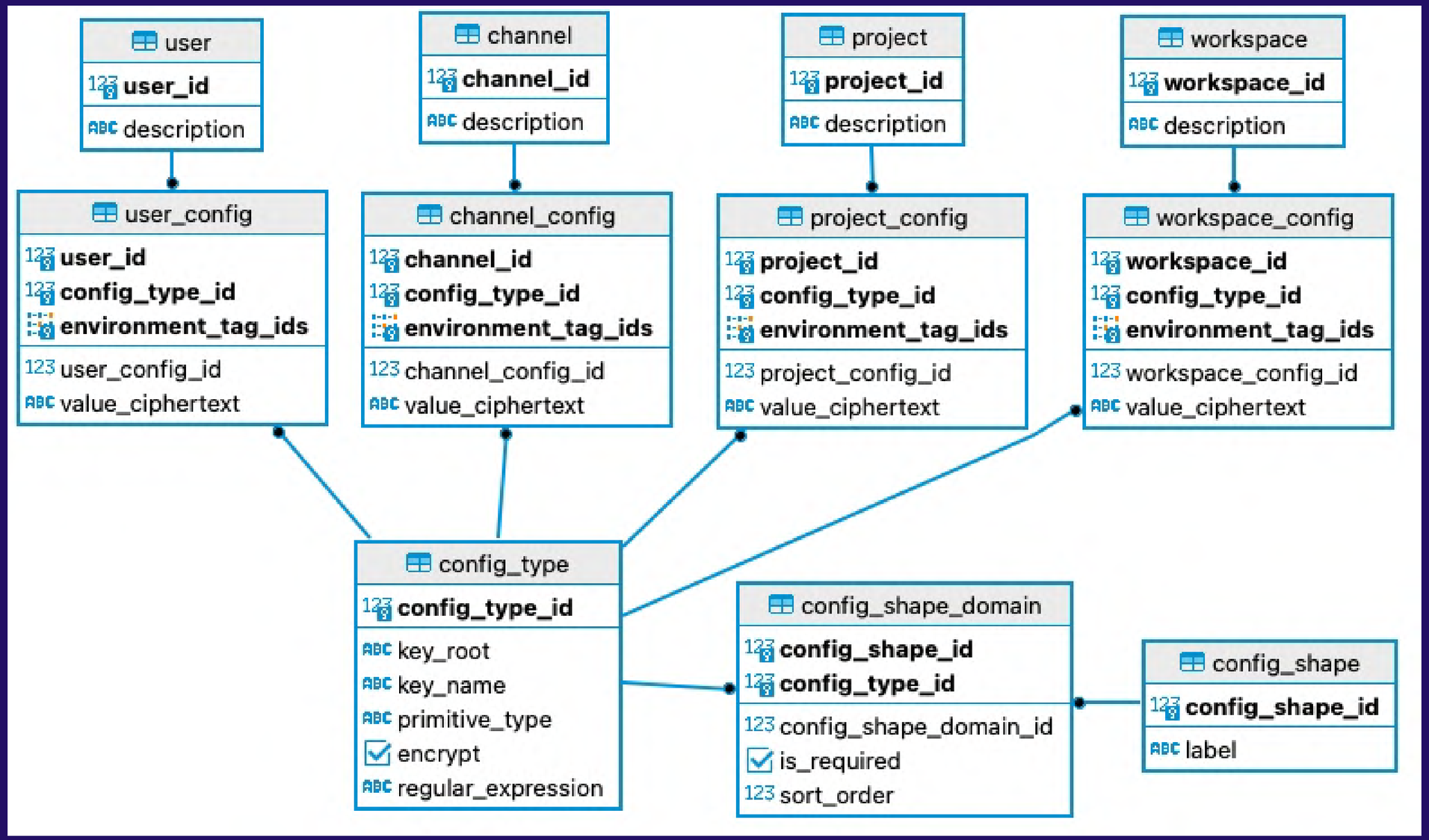
Pitch

Entity Attribute Value (EAV) model

A data model for situations when a number of different attributes can be used to describe an object, but only few attributes actually apply to each one.

Pitch

Database Design



Pitch

Active Record Implementation

Pitch

```ruby
class ConfigType < ActiveRecord::Base
  validates_presence_of :key_name, :primitive_type
  validates_uniqueness_of :key_name

  # @param value [String] Value to deserialize
  def deserialize(value)
    ConfigTypeDeserializer.call(value)
  end
end
```

Pitch

```ruby
class ChannelConfig < ActiveRecord::Base
  belongs_to :channel
  belongs_to :config_type

  validates_presence_of :value, :environment_tag_ids

  # Encryption/decryption for RoR using Lockbox
  encrypts :value
end
```

Pitch

```ruby
class Channel < ActiveRecord::Base
  has_many :channel_config
end
```

Pitch

```
email = Channel.find(1)
ChannelConfig.where(channel: email)

[
  #<ChannelConfig id: 1, config_type_id: 1, channel_id: 1, value: [FILTERED]>,
  #<ChannelConfig id: 2, config_type_id: 2, channel_id: 1, value: [FILTERED]>,
  #<ChannelConfig id: 3, config_type_id: 3, channel_id: 1, value: [FILTERED]>,
  #<ChannelConfig id: 3, config_type_id: 4, channel_id: 1, value: [FILTERED]>,
]
```

Arrays are cool. Know what's cooler?

Hashes for hierarchical data.

We could create a function to transform arrays into hashes, but...

Pitch

We are Rubyists - we like DSLs :)

Pitch

"Elegance is when the inside is as beautiful as the outside"

Coco Chanel

Pitch

DSL for Lookup

```
email = Channel.find(1)
email_config = ConfigAdaptor.hashify do
  has_many email.channel_config
  symbolize_keys true
end

# {
#   :sender_name=>"MagicBell Notifications",
#   :sender_email=>"hello@magicbell.io"
#   :api_key=>"dPj4N7A5agstyHye",
# }
```

Pitch

- The Docile gem makes creating the DSL a breeze
- Great DX for your team
- Works with any object that has config (using config_type)

```ruby
class ConfigAdaptor
  def has_many(has_many)
    @has_many = has_many
  end

  def hashify
    @has_many.map do |config_item|
      config_type = config_item.config_type
      key = config_type.key_root.to_sym
      value = config_type.deserialize(config.value)

      [key, value]
    end.to_h
  end

  def self.hashify(&block)
    Docile.dsl_eval(ConfigAdaptor.new, &block).hashify
  end
end
```

Pitch

Merge for free!

```
email_config =
  email_channel_config
    .deep_merge(workspace_config)
    .deep_merge(project_config)
    .deep_merge(user_config)
```

Pitch

But, does it scale?

Pitch

45 million

Configuration entries
with Environment and Entity precedence

3ms

Fetch

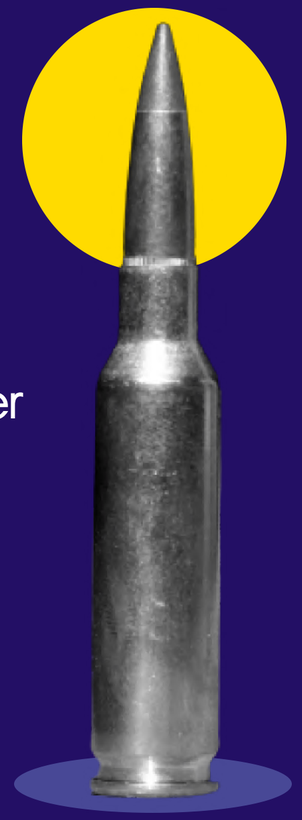
Benchmarks data at

https://github.com/magicbell-io/eva-config

Pitch

Not A Silver Bullet

- Complexity - a high learning curve for new developers.
- Loose data validation that needs to be implemented later in the code.



Pitch

What's next

- Extract it to a Gem
- Create generators for RoR
- Create materialized views to improve performance when needed

Pitch

Come, work with us!

Like this, or hate? Come help us improve it :)

hello@magicbell.io

Pitch