This slide intentionally left blank

Arbitrary multi-tenant configuration for ~~fun and~~ profit

A “Yet Another Gem Someone Made and Wants to Show People” talk

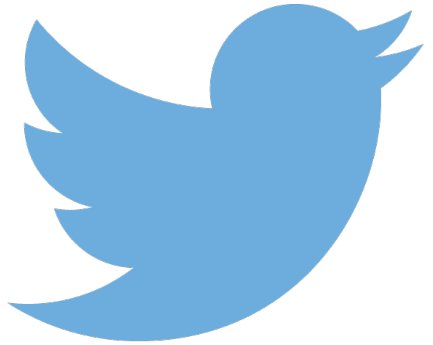
indy.rb – September 2016

Who am I?

I’m Tony

t27duck

t27duck



@t27duck

MOBI®

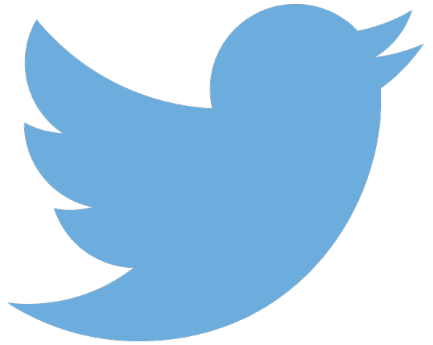
Senior Web Developer

GONINTENDO

Web Master / Server Admin

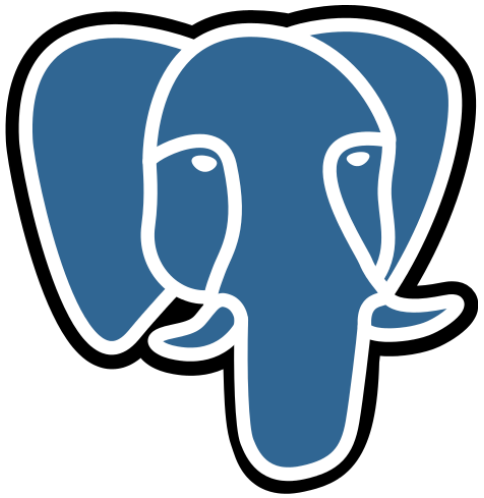
t27duck

t27duck



@t27duck

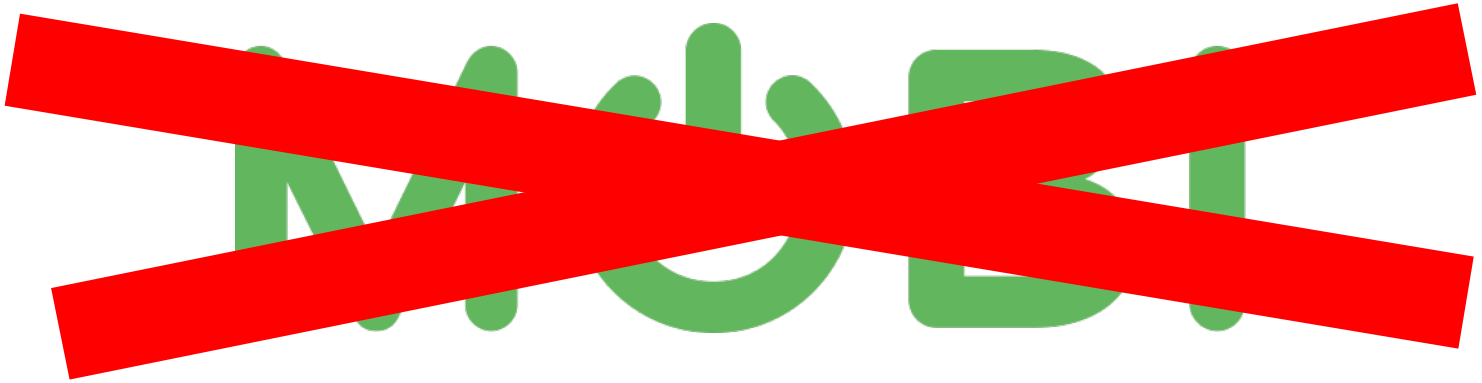
My likes…



My bae

Best game ever

Let’s talk about a business with a problem…



IBOM

The Universe

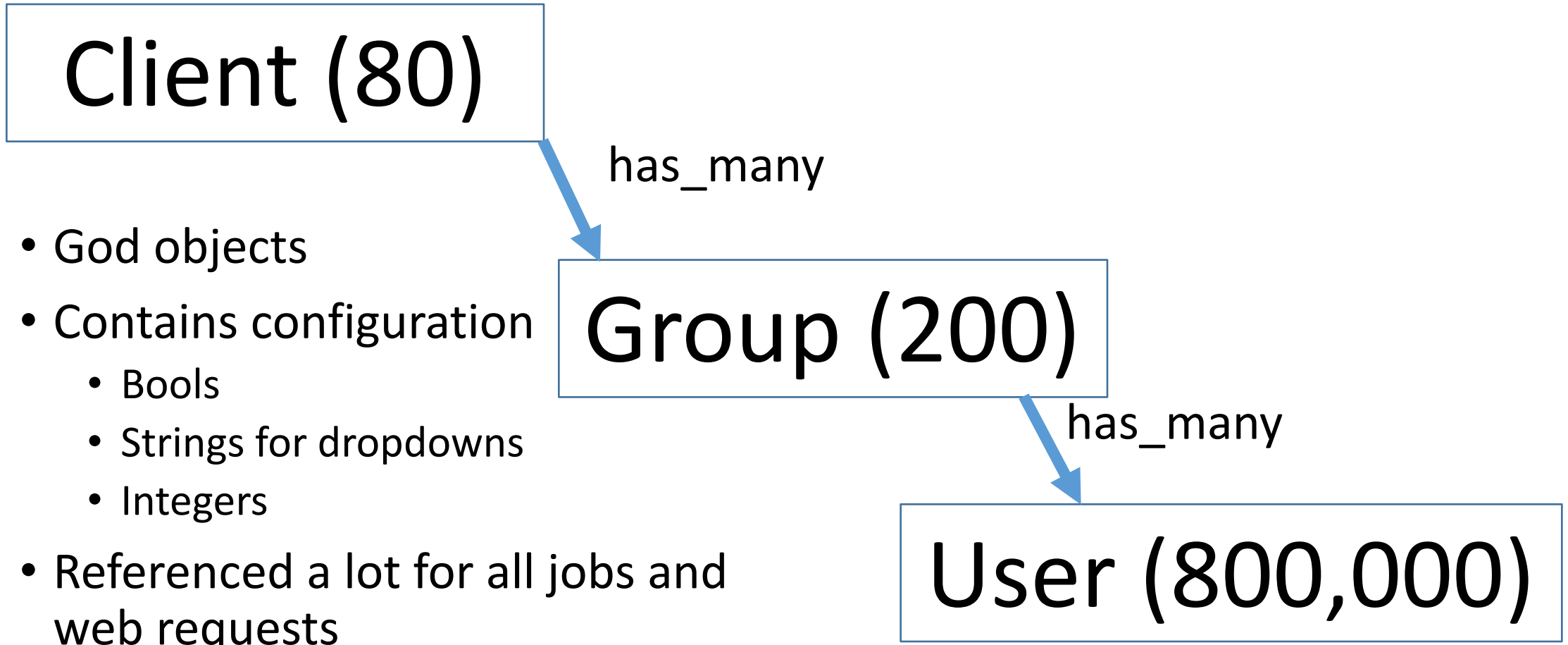
IBOM

- Manages Apple™ adaptors for corporate clients (#courage)
  - Procurement
  - Expense optimization
- > 80 Clients
  - 24/7 traffic (Heaviest during US time)
  - “Global”
- Diverse user base
  - Administrators
  - Operations
  - End users

- 2 - 8 production deploys / week
  - #shipit
- ~40 unicorn workers
- ~10 resque workers
  - Long running imports and exports of large amounts of data throughout the day and evening
- PostgreSQL
- ~25 developers
- Prefers **configuration** over **customization** for new features

The Universe

IBOM



- God objects
- Contains configuration
  - Bools
  - Strings for dropdowns
  - Integers
- Referenced a lot for all jobs and web requests

The Scenario

- New feature to be added
- Needs to be configurable as not all clients want/need it

add_column :clients, :new_feature, :boolean

or

add_column :groups, :new_feature, :boolean

- Migration runs in < 0.02 seconds locally

The Problem

IBOM

- Deploy during the day == App goes down
- Deploy hangs during migrations
- Requests queue up on unicorn
- Background jobs sit there
- Exceptions when migration finally completes

- … but all we did was add a bool to a “small” table

The Problem – Table locking

- Altering these tables is no small feat
  - ALTER TABLE with a null column still requires an exclusive lock (though it's "quick" since null is the default)
  - If setting a default value, the entire table must be rewritten on disk (longer lock)
- Long running jobs, especially ones that run in a transaction, blocks the migration until the transaction commits

The Problem – App Exceptions

- Adding a column to a constantly accessed table will commonly throw an exception for a web request during the migration
  1. Process in a transaction starts (Background job or an ActiveRecord save on a model with hooks referencing the modified table)
  2. Structure of the table changes
  3. Cached prepared statements in ActiveRecord are now invalid
  4. PostreSQL throws an error about the invalid statement on the next query in the transaction regarding the altered table
  5. ActiveRecord rescues error and refreshes statement cache
  6. Because the error was thrown in a transaction, PostgreSQL makes the entire transaction invalid
  7. Exception is thrown – ERROR: current transaction is aborted, commands ignored until end of transaction block

Original Issue: https://github.com/rails/rails/issues/12330

Claimed to be 'Fixed' in Rails 5: https://github.com/rails/rails/pull/22170

Possible Solutions

- Pause workers before deploy
  - “After hours”
  - Doesn’t solve app exceptions for existing users at the wrong place at the wrong time
  - Requires team coordination
- Deploy during the “early hours” of the morning
  - Requires team coordination
- Relegate deploys to known “maintenance windows”
- Stop developing features

“There’s got to be a better way”™

Dude using a hammer to put lettuce in a food processor/blender

has_config

https://github.com/t27duck/has_config

(Disclaimer: I made this)

Elevator Pitch

- Add one column to a table
  - Hash (serialized attribute)
  - JSON/JSONB (PostgreSQL data type)
- In the model, list out the configuration items
  - Name
  - Data type
  - Optional default
  - Optional validations

- Move away from adding column after column to tables just for configuration
- Requires minimal (if any) changes to existing code (including forms)
- To the developer, it feels like you're just working with a regular attribute (including a setter method)
- Still query-able (when used with the JSON data type)

An Example

```
class AddConfigurationToClients < ActiveRecord::Migration
  add_column :clients, :configuration, :text
end


class Client < ActiveRecord::Base
  serialize :configuration, Hash
  include HasConfig::ActiveRecord::ModelAdapter
  has_config :primary_color, config: { type: :string, default: 'green' }
  has_config :secondary_color, config: { type: :string }
  has_config :rate_limit, config: { type: :integer, validations: { numericality: { only_integer: true } } }
  has_config :category, config: { type: :string, validations: { inclusion: { in: CATEGORIES } } }
  has_config :active, config: { type: :boolean, default: false }
end
```

Also available at https://github.com/t27duck/has_config/blob/master/README.md

So what does this give us?

- Getter and setter method for each configuration
  - Treat form helpers as if they were normal attributes*
  - Treat I18n as if they were normal attributes
  - Treat everyday code as if they were normal attributes
- A “?” method for bools
- Rails validations
- Default values

* Manually declare control type if library normally infers based on attribute

Playing in `rails c`

```
client = Client.new

client.default_color
=> "green"

client.secondary_color
=> nil

client.active
=> false

client.active?
=> false
```

```
# Like if this was from a form
client.active = '1'
=> '1'

client.active?
=> true

client.rate_limit = 3
=> 3

client.valid?
=> false

client.errors.full_messages
=> ["Category is not in the list"]
```

Also available at https://github.com/t27duck/has_config/blob/master/README.md

Other features

- Chaining through related configurations
  - Group as config “foo”
  - Client as config “foo”
  - Ask a group for the “foo” configuration, defer to client’s ”foo” if blank
- Configuration file
  - Similar to delarative_auth’s authroization_rules.rb file

More info available at https://github.com/t27duck/has_config/blob/master/README.md

Future features

- “Spring cleaning” method to remove unknown config keys
- Method to migrate from existing column to configuration hash
- Use Rails 5 ActiveModel Attributes API?
- Support hstore

I accept pull requests! ☺

Pros

- We add one column and add on as many new configuration items we want as we need them
- No need for additional columns
- Still query-able via JSON
  - SELECT configuration->’theme’ FROM clients;

Cons

- Relies on Ruby some meta-programming magic
  - Could be confusing for newer developers at first glance
- Need to be cognizant of what we're marking as configuration - we may still want a new column for some cases
- Not the end-all-be-all solution
- Old "junk" configurations could linger in rows

So then why this gem?

- Think of has_config not as a bandage, but more like a duct tape
  - Helps solve the immediate problem
  - Relatively straight-forward
  - Allows us to add configuration options at will
- Gets us back to coding and not worrying about adding another column and killing the app
- Got no better ideas at the moment

The End!

https://github.com/t27duck/has_config

Gets these slides at https://github.com/t27duck/showandtell

Support the Ruby Gem Infrastructure!

https://rubytogether.org/