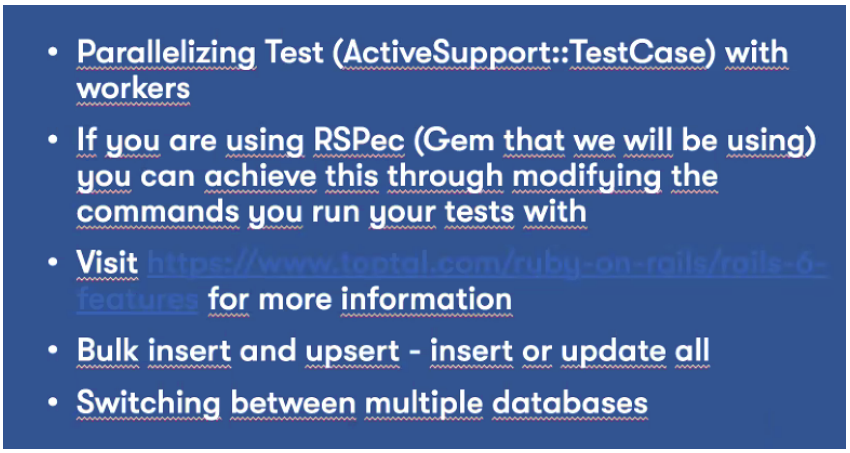
New features from version 6



Directories

- Log
  - Application log files
- Test
  - Unit tests and fixtures
- Rakefile
  - Loads tasks tun from the CLI
- tmp
  - Temporary files like the pid
- storage
  - Active storage files for disk service
- vendor
  - Downloaded rubies

Active ORM

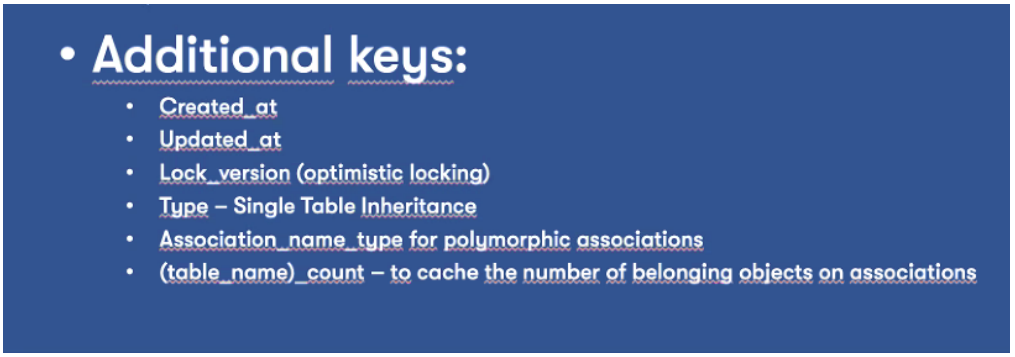
- Connects rich objects to tables in the db.
- Represents models and their data in the ruby language
- Represents associations
- Represents inheritance hierarchies
- validates models
- Performs database operations in an Object oriented fashion.

Conventions

- Model class -> Singular in each word and capitalized
- Database -> plural with underscores

Schema conventions

- Foreign keys -> singuralarized_table_name_id
- Primary key -> use id and bigint for postgreSQL and MySQL or integer for SQlite



Creating Active records models

Inherit from ApplicationRecord, which inherit from ActiveRecord::Base

```ruby
class Article < ApplicationRecord
  include Visible

  has_many :comments, dependent: :destroy

  validates :title, presence: true
  validates :body, presence: true, length: { minimum: 10 }
end
```

Validations are not stored on the database.
Having the validations on the database ?
Things like length of text maybe on a database.

Custom primary key

```ruby
class Product < ApplicationRecord
  self.primary_key = "product_id"
end
```

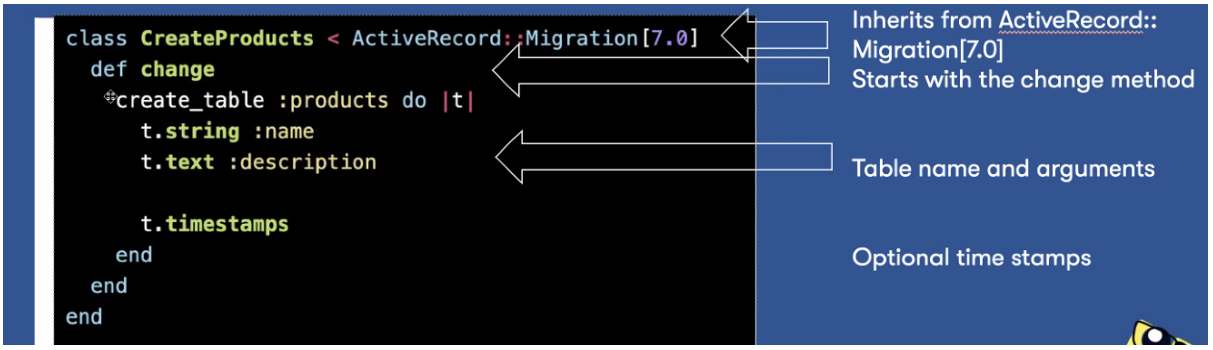
Active record migrations

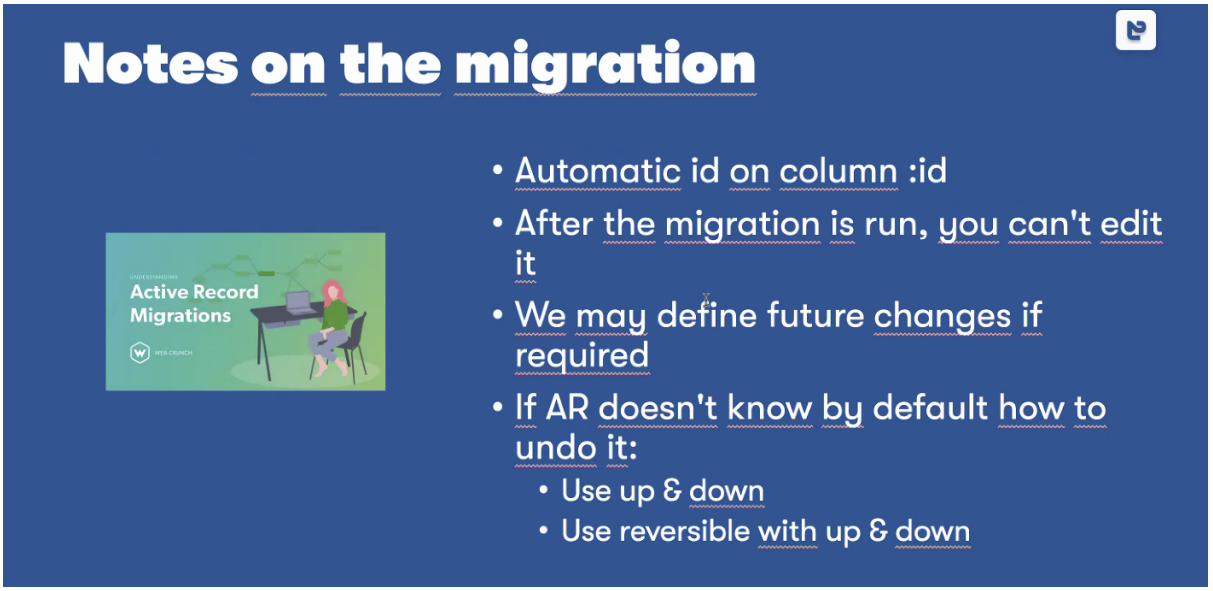
Feature that allows the database to evolve over time.

Database schema

You can see your database at schema.rb.
Never modify it.

Example of a migration





Not a good practice to update a migration.

Using up and down

- Up
    - what happens when migrating
- Down
    - What happens when rolling banking

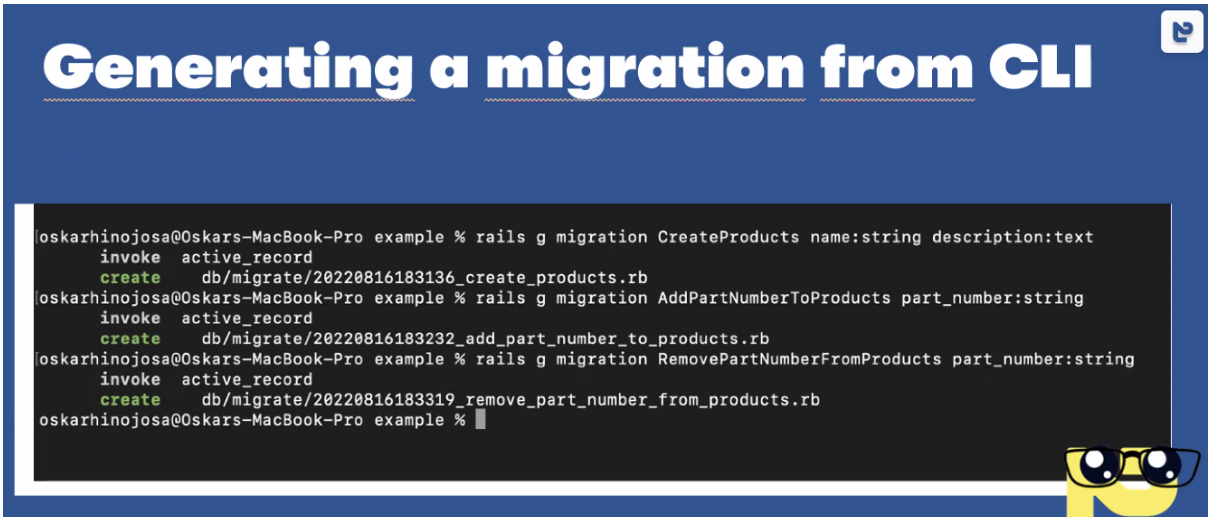
```ruby
class ChangeProductsPrice < ActiveRecord::Migration[7.0]
  def up
    change_table :products do |t|
      t.change :price, :string
    end
  end

  def down
    change_table :products do |t|
      t.change :price, :integer
    end
  end
end
```

When to use the rollback?
When you need to clear the db. And you have to delete all the test instances of data. Things you use to develop.

Probably you make a mistake and you cannot go back to the last commit.
It is going to be better to do a rollback.

Instead to do a whole migration, you can create a migration to rollback an specific migration that you did.



Example of files generated

```ruby
class AddPartNumberToProducts < ActiveRecord::Migration[7.0]
  def change
    add_column :products, :part_number, :string
  end
end
```

```ruby
class RemovePartNumberFromProducts < ActiveRecord::Migration[7.0]
  def change
    remove_column :products, :part_number, :string
  end
end
```

Creating JOIN tables

- A product that may have multiple customers and a product that may have multiple products

```
oskarhinojosa@Oskars-MacBook-Pro example % rails g migration CreateJoinTableCustomerProduct customer product
      invoke  active_record
      create    db/migrate/20220816184358_create_join_table_customer_product.rb
oskarhinojosa@Oskars-MacBook-Pro example %
```

Run migration
rails db:migration

- $ rails db:migrate [VESION=20220822162]

db:migrate:down or db:migrate:up if you want to run those commands on specific.

Up happens when migrating.
Down runs on rollback.

Dropping a column is not reversible by default.
If you include information like foreign keys, etc. there is no problem.
If you don't, you will have an active record error.

Some operations are not reversible by default unless you give more information to the command.

If you don't use down and up and you delete a column. It will not know what data that column had when you revert it.