

ORM (Object Relational Mapping)

Advantages

- ⊕ DB independent. moving from PG to Sqlite should be trivial.
- ⊕ Cleaner code focused on business logic rather than SQL Query
- ⊕ Relationship/Association Management
- ⊕ Model Objects can be cached -- No need to repeatedly query DB for same data
- ⊕ Transaction support
- ⊕ Built in validation before save

Disadvantages

- ⊕ Abstraction = overhead (uses more memory etc...)
- ⊕ SQL ignorance

Create Table (with Datatypes)

```
# CREATE
class CreateOrders < ActiveRecord::Migration
  def change
    create_table :orders do |t|
      t.belongs_to :category
      t.binary :photo # PG: bytea MySQL:
      blob
      t.olean :success # PG: boolean
      MySQL: tinyint(1)
      t.date :ship_date # PG & MySQL: date
      t.datetime :order_date # PG:
      timestamp MySQL: datetime
      # PG & MySQL: decimal Precise. Use
      for currency, geolocation
      t.decimal :latitude, :decimal,
      :precision => 15, :scale => 13
      t.float # PG & MySQL: float Less
      precise than decimal
      t.integer :age # PG: integer MySQL:
      int(11)
      t.string :name # PG & MySQL:
      varchar(255)
      t.text :comments # PG & MySQL: text
      t.time :ship_time # PG & MySQL: time
      t.timestamps # PG: timestamp
      MySQL: datetime
    end
  end
end
```

Generators

```
rails g model comment post:references text:string
```

```
rails g migration add_post_to_user post:belongs_to
```

```
rails g migration AddUserRefToProducts user:references
```

```
rails g migration AddPartNumberToProducts part_number:string
```

```
rails generate controller Sessions new
```

rails db: command

Rollback the last migration:

```
rails db:rollback
```

Revert the last 3 migrations:

```
rails db:rollback STEP=3
```

Rollback and then migrating back up:

```
rails db:migrate:redo STEP=3
```

Run specific migration

```
rails db:migrate:up VERSION=20080906120000
```

Load test fixtures

```
rails db:fixtures:load
```

Create a db/schema.rb file that can be portably used against any database supported by ActiveRecord

```
db:schema:dump
```

Load a schema.rb file into the database

```
db:schema:load
```

Dump database structure to SQL file Drops the database, creates the datab

```
db:structure:dump
```

Clone your database structure into the test database

```
db:test:prepare
```

Drops the database, creates the database and then runs migrations against the database. Takes a VERSION argument as well as RAILS_ENV

```
db:reset
```

Create/clear Sessions table

```
db:sessions:create (or :clear)
```

Postgres Specific Data Types

```
:hstore # Key => Val Hash Store
:json #The json data type stores an exact copy of
the input text
:jsonb # stored in a decomposed binary format
:array
:cidr_ address #IP Address
:ip_ address #IP Address
:mac_ address #String
```

In other DBs these are all stored as strings

Change Table

```
class AddPartNumberToProducts <
ActiveRecord::Migration[5.0]
  def change

    add_index :products, :part_number
    remove_index :products, :part_number
    add_reference :this_table, :other_table, index: true
    remove_reference :this_table, :other_table, index: true

    belongs_to:
    add_column :products, :part_number,
:string
    add_column :products, :column_name,
:string, :limit => 50
    rename_column :shoes, :season, :season_id
    change_column :column_name, :new_column_type
    change_column_default :table_name,
:status, 0
    remove_column :column_name

  end
end
```



By dwapi
cheatography.com/dwapi/

Not published yet.
Last updated 8th October, 2017.
Page 2 of 2.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>