Reporting on Rails
ActiveRecord and ROLAP Working Together

Tony Drake

RailsConf 2017

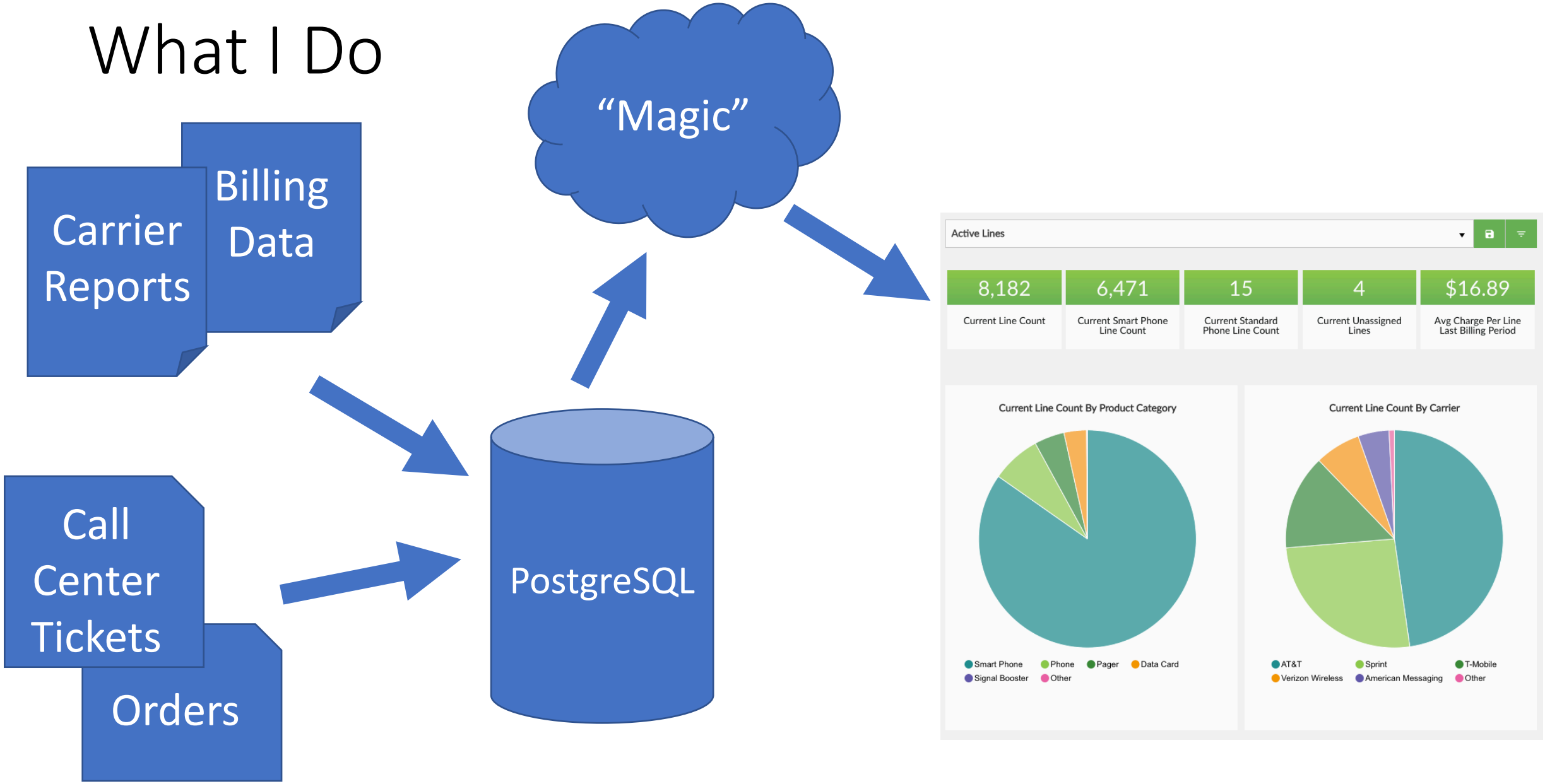
github.com/t27duck

@t27duck

Who am I?

- Tony Drake
- Senior Developer at MOBI
  - Billing and Reporting Team
  - Nearly one million devices under management + Billing data
- Seven Years Working with Rails
- Mario Kart Connoisseur

What I Do


"Magic"
Billing Data
Carrier Reports
Call Center Tickets
Orders
PostgreSQL
Active Lines
8,182
Current Line Count
6,471
Current Smart Phone Line Count
15
Current Standard Phone Line Count
4
Current Unassigned Lines
$16.89
Avg Charge Per Line Last Billing Period
Current Line Count By Product Category
Smart Phone
Phone
Pager
Data Card
Signal Booster
Other
Current Line Count By Carrier
AT&T
Sprint
T-Mobile
Verizon Wireless
American Messaging
Other

The Scenario

- Feature Request: Build a series of dashboards to **report** on our data
  - Dashboard for client administrators
  - Dashboard for internal support staff
  - Dashboard for MOBI Management
- Allow for user-defined filtering

- *Where do you begin?*

Couple notes…

- Don’t want “bloat” in our result set
- No ActiveRecord instances (Less memory)
- Generic, uniformed data (Arrays of rows)
- Only get the information we care about

What is “reporting”?

- Data is stored in our data stores (RDBMS)
  - Worthless to humans
  - Useful to computers
- What’s useful? *Information*
- Reports turn data into information for humans
- Reports answer specific questions

What do you want to answer?

- For internal support staff (Workflow ease)
  - How many support tickets by client were created in the last month?
- For client administrators (Visibility into program)
  - What is the sum of mobile charges for for the last billing period by cost center?
- For MOBI Management (Company health)
  - How many active lines of service by client?

What is “OLAP”?

- **O**n**l**ine **A**nalytical **P**rocessing
- “…the technology behind many Business Intelligent (BI) applications. OLAP is a powerful technology for data discovery, including capabilities for limitless report viewing, complex analytical calculations, and predictive “what if” scenario planning.”

From: http://olap.com/olap-definition/

- Data is organized into “data cubes”
  - Comprised of “dimensions” and “measures”
  - Every combination known and calculated ahead of time
- Large amounts of preprocessed data stored in a warehouse
- Commonly deals with aggregate data (count, max, min)

What is “**R**OLAP”

- **R**elational **O**n**l**ine Analytical **P**rocessing
- OLAP functionality implemented with a RDBMS
- **Dynamic queries** generated for reports
- Uses standard database **tables** and **relations**
- May be implemented on both **transactional** data (OLTP) and **warehouse** data

OLAP Terminology

- Fact Table (Sometimes called “Fact Model”)
- Dimension
  - Members (Labels)
  - Hierarchy
- Dimension Filters (also known as just “Filters”)
- Measure
- Metric

Fact Table/Model

- The primary table where information is derived from in a report
  - Fact columns – Commonly numeric columns
  - Dimension columns – values that may be grouped together or references other tables

SQL: FROM clause

Rails: ActiveRecord model

How many support tickets by clients were created in the last month?

How many active lines of service do we support broken down by client?

Dimension

- A point in the data where you can ”slice and dice” fact model info
  - Carrier
  - Cost center
  - State of an order in a state machine
- Lives on fact table or as a foreign key to another table

SQL: JOIN, GROUP BY

Rails: ActiveRecord relation or attribute

How many open support tickets are in my queue by type?

What is my active lines of service count by carrier?

Dimension Hierarchy

- Related attributes on a dimension used to “drill up” and “drill down”
- Found on dimensions which are relations to a fact model

Examples:

- Dates: Date, Month, Quarter, Year
- Mobile Phone: Model, Manufacture, OS, Wireless Technology

Dimension Members (Dimension Labels)

- Information related to a dimension
- When on fact table, the label is the column
- When on a relation, a field representing the hierarchy level

Dimension Filters (or just “Filters”)

- Not a “real” OLAP term
- Takes advantage of querying capabilities of RDBMS
- Allows for more fine-grained reporting

SQL: WHERE

Rails: where(), scopes, ransack

Measure

- A column in a table (usually numeric) used in aggregations
  - Average, Sum, Maximum, etc
- Examples:
  - Total amount in a sale
  - Number of units used in a transaction

SQL: A column in a fact table

Rails: ActiveRecord attribute

Metric

- A measured value; The subject of the report
- The thing you actually want to answer

SQL: The query

Rails: All the things

What is the sum of charges for the last billing period by cost center?

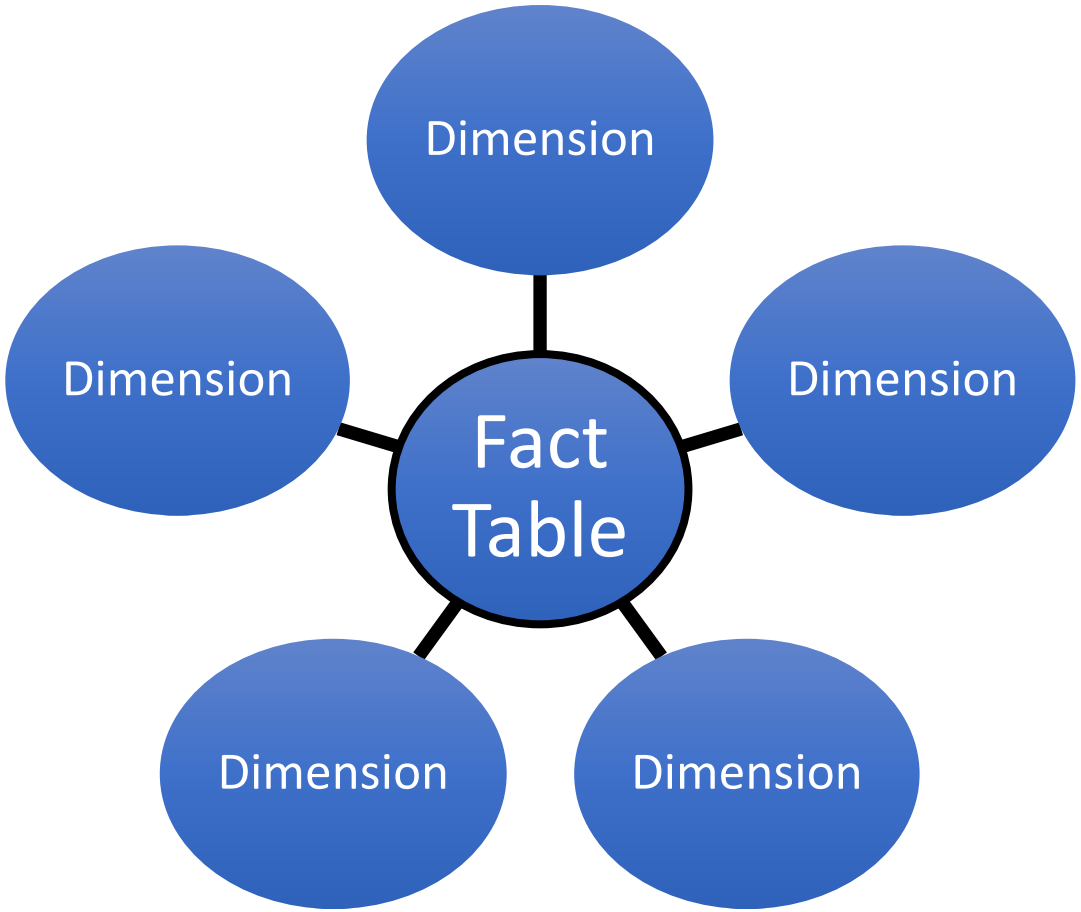
How many support tickets by client were created in the last month?

Terminology

ROLAP	SQL	Rails
Fact Table	FROM	ActiveRecord Model
Dimension	JOIN, GROUP BY	AR Relations, joins(), group()
Dimension Filter	WHERE	Scopes, where(), ransack
Measure	Numeric Column	Model Attribute
Metric	Query	All the above

What is the sum of mobile charges for the last billing period grouped by cost center?

Star Schema

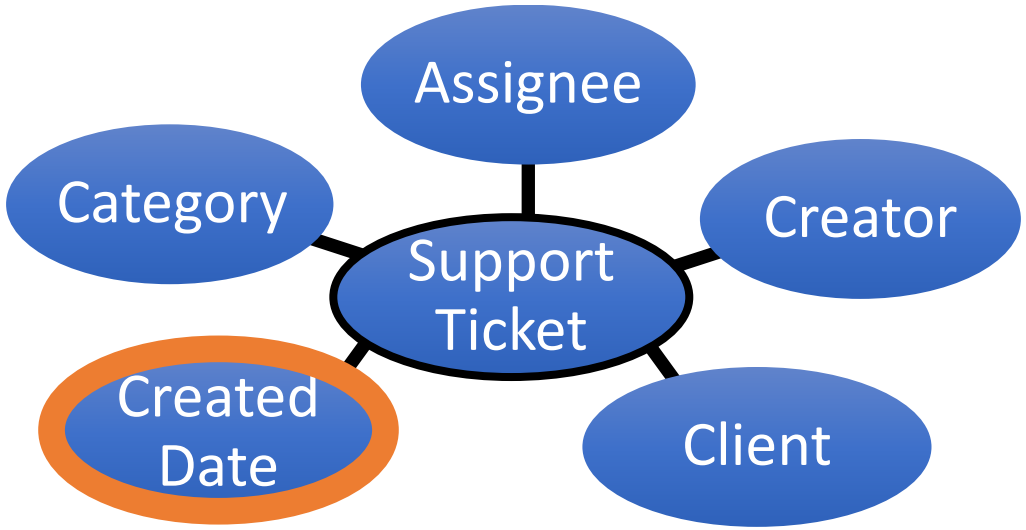


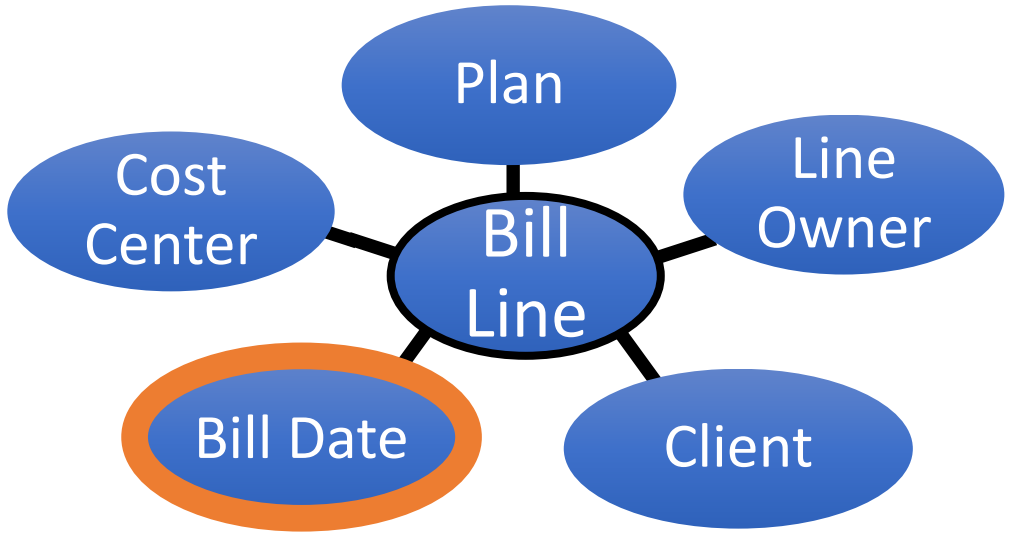
- Design pattern for organizing data in a data warehouse
- Consists of measures and dimensions that live on the fact table
- belongs_to / has_one branch out to relations via foreign keys
- **DOES NOT SUPPORT has_many relationships (well)**
- Other option: Snowflake Schema

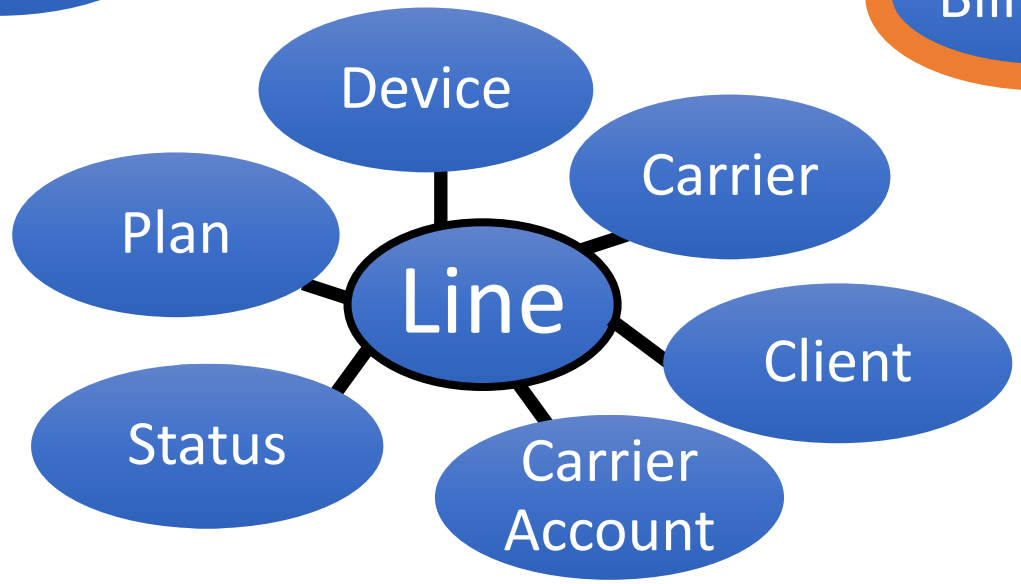
Star Schema

MOBI®

Overly simplified examples





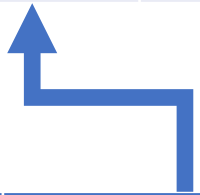


Dates as Dimensions

- Want to report by Quarter? Year? Month?
- Date functions on date columns can't use a regular index
- Simple join + group/filter is quicker

date_dimensions

id	date	mday	month	year	quarter	wday
20170509	2017-05-09	9	5	2017	2	3
20170510	2017-05-10	10	5	2017	2	4
20170511	2017-05-11	11	5	2017	2	5
20170512	2017-05-12	12	5	2017	2	6



support_tickets

id	created_at_id	client_id	assigned_id_to	state	category
16	20170509	123	15	….	….
17	20170510	342	90	….	….
18	20170511	123	15	….	….
19	20170512	586	76	….	….

Great! ActiveRecord can do all that… right?

- ActiveRecord's internals can provide all the information needed to *construct* ROLAP *queries*
- Relationship information (joins and grouping)
- Filtering capabilities (Scopes, where(), ransack)
- Ability to select out specific columns

Great! ActiveRecord can do all that… right?

- No programmatic way to easily group by all non-aggregate columns
- Aggregation methods (#count, #maximum, #minimum) do not allow for full control over multiple columns returned
- No way to describe a fact table or metrics in ROLAP terms
- No decent way to defining what a user can filter metrics on

Options?

- Manually write hardcoded queries?
- Write a queryer yourself?
- Switch your application to sequel?

active_reporting

- https://github.com/t27duck/active_reporting
- Implements a DSL for describing fact models, dimensions, and filters
- Uses ActiveRecord to build a query and execute it on the database
- Does not dirty up ActiveRecord (only one new method)
- *Mostly* production-ready
  - API pretty much at a good spot
  - Would love help with documentation :D

active_reporting – FactModel

```
class LineFactModel < ActiveReporting::FactModel

end
```

```
class Line < ActiveRecord::Base

end
```

active_reporting – Dimensions

```
class LineFactModel < AR::FM
  …

  dimension :number
  dimension :carrier
  dimension :some_column

  …
end
```

- Define specific columns or relations for dimensions
- Relation-based dimensions include identifier column and label in report results

active_reporting – Heirarchy + Labels

```
class DateDimFactModel < AR::FM
   …

 default_dimension_label :date

 dimension_hierarchy [
  :date, :month, :year, :quarter
 ]
   …
end
```

- Default label is “name”
- Defining a hierarchy allows for specifying different columns to group by while dimensioning

active_reporting – Dimension Filters

```
class TicketFactModel < AR::FM
  …
  dimension_filter :scope_on_model

  dimension_filter :by_creator_id,
       ->(x) { where(creator_id: x) }

  dimension_filter :subject_cont,
     as: :ransack
  …
end
```

- Define available filters
- Scopes defined on the ActiveRecord Model
- Defined just for the fact model using scope syntax
- Optional way to fallback to ransack on the AR model

active_reporting – Metric

```
m1 = ActiveReporting::Metric.new(
    :line_count_by_carrier,
    fact_model: LineFactModel,
    dimensions: [:carrier]
)

m2 = ActiveReporting::Metric.new(
    :total_charges,
    fact_model:   BillLineFactModel,
    measure:      :total_charges,
    aggregate:    :sum
)
```

- Describes a question to answer
- Declare
  - Fact Model
  - Dimensions
  - Measure
  - Aggregate (defaults to count)

active_reporting – Report

```
metric = ActiveReporting::Metric.new(
    :total_charges,
    fact_model:   BillLineFactModel,
    measure:      :total_charges,
    aggregate:    :sum
)

r = ActiveReporting::Report.new(
    metric,
    dimensions: [:carrier],
    dimension_filter: {
      carrier_id_eq: [123, 456]
    }
)
```

- Builds and executes the report
- Takes a pre-build metric and expands on it
- Add additional dimensions and filters (ie, user input)
- Returns simple array of hashes

active_reporting – Report

```
metric = ActiveReporting::Metric.new(
    :total_charges,
    fact_model:   BillLineFactModel,
    measure:      :total_charges,
    aggregate:    :sum
)

r = ActiveReporting::Report.new(
    metric,
    dimensions: [:carrier],
    dimension_filter: {
      carrier_id_eq: [123, 456]
    }
)
```

```
SELECT
  SUM(bill_lines.total_charges)
    AS total_charges,
    carriers.id AS carrier_identifier,
    carriers.name AS carrier
FROM bill_lines
JOIN carriers
      ON carriers.id = bill_lines.carrier_id
WHERE bill_lines.carrier_id IN(123, 456)
GROUP BY carriers.id, carriers.name
```

active_reporting – Report

```
SELECT
  SUM(bill_lines.total_charges)
    AS total_charges,
    carriers.id AS carrier_identifier,
    carriers.name AS carrier
FROM bill_line
JOIN carriers
      ON carriers.id = bill_lines.carrier_id
WHERE bill_lines.carrier_id IN(123, 456)
GROUP BY carriers.id, carriers.name
```

```
> r.run
=> [
  {
    total_charges: 742.34,
    carrier_identifier: 123,
    carrier: 'AT&T'
  },
  {
    total_charges: 432.34,
    carrier_identifier: 456,
    carrier: 'Sprint'
  },
]
```

Other Database Considerations

- Try not to have to make “multiple jumps” to tables
  - has_one :through can let us “cheat”, but query isn’t necessarily optimal
  - Keep dimensions “one deep” when focusing on Star Schema
- Rails counter caches
- Pre-calculated aggregates (Rebuilt via background jobs)
- Index liberally where needed
  - Foreign keys
  - Columns commonly used for filtering
  - Use EXPLAIN [ANALYZE]
- Read-only replicating slaves
- Shard or schema separation

The End

- ActiveReporting Gem: https://github.com/t27duck/active_reporting
- Slides: https://github.com/t27duck/showandtell
- Twitter: @t27duck