

This slide intentionally left blank

# Reporting on Rails ActiveRecord and ROLAP Working Together

**WIP Edition**

Tony Drake

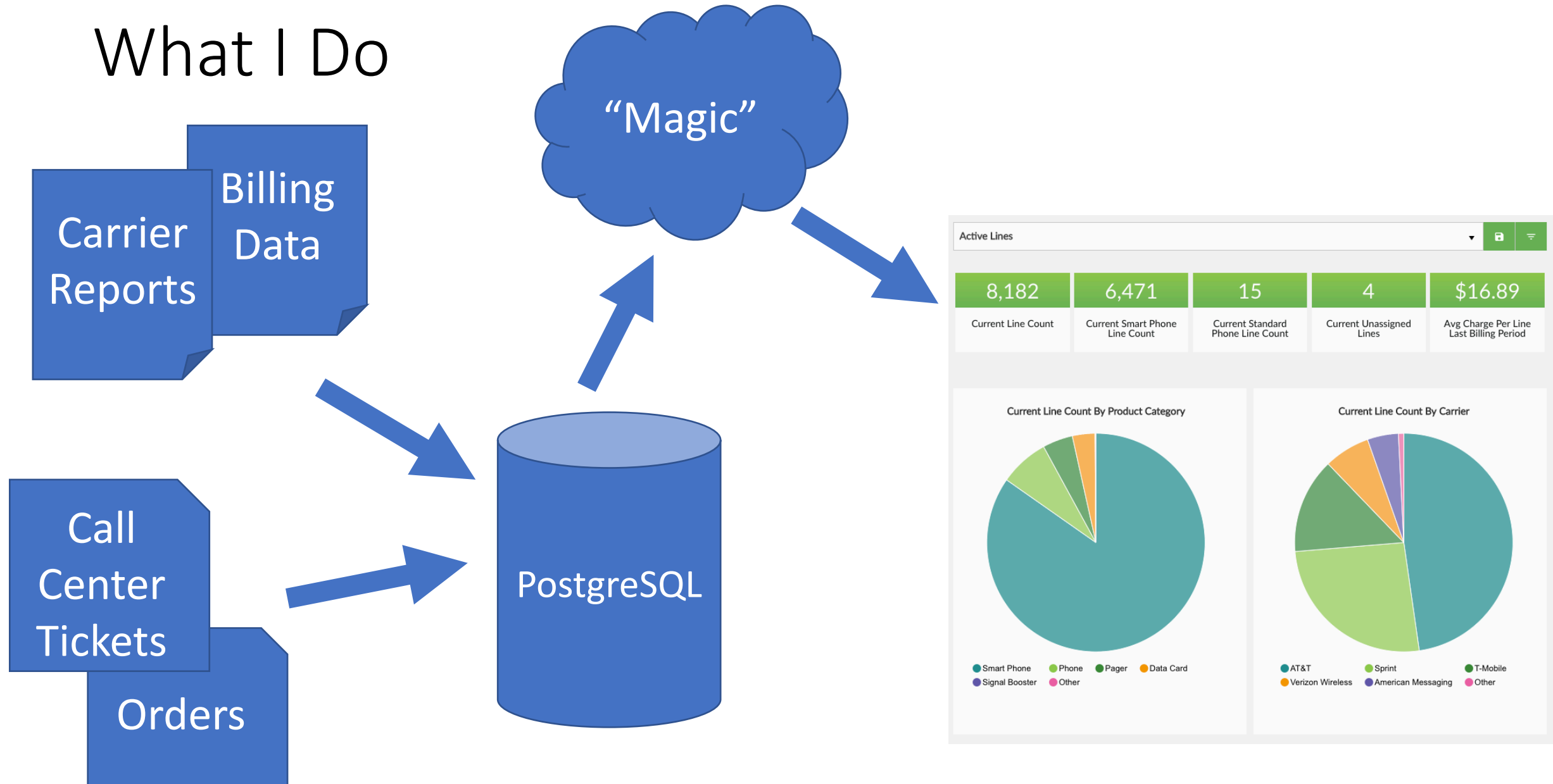
IndyRB March 2017

# Who am I?

- Senior Developer at MOBI Wireless Management
  - Billing and Reporting Team
  - "Nearly one million devices under management." ~ Our Marketing
- Seven Years Working with Rails
- Mario Kart Connoisseur



# What I Do



# Today's Talk

- Define “reporting”
- Industry standard terminology for “reporting”
- Ways of organizing your data for “reporting”
- How much can Rails do out of the box?

# The Scenario

- You are: Lead Developer
- Median
  - Semi-successful blog site
  - ~60,000 registered users
  - ~1 million blog entries
  - ~1.8 million comments

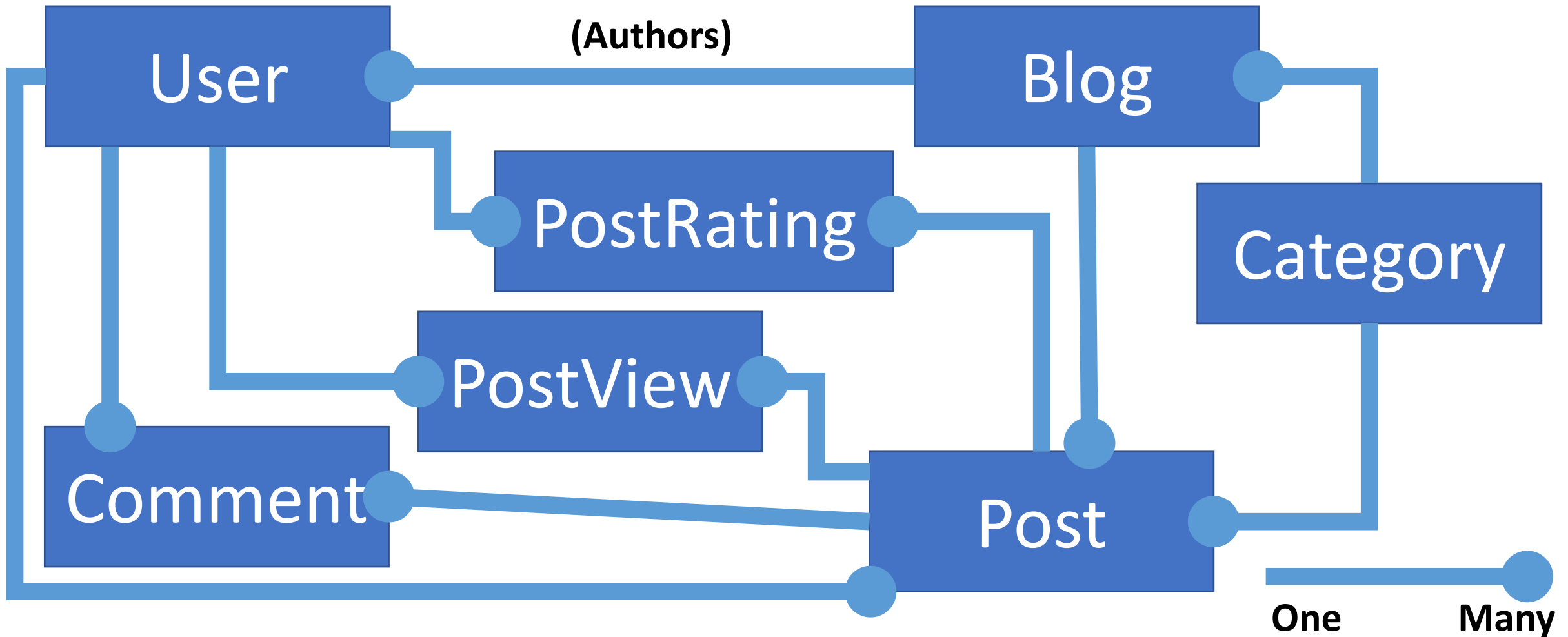


*Your Voice as we Travel the Highway of Life*

(I spent way too much time on this fake logo)

# The Scenario

Just a rough estimate / example.  
Don't bother memorizing this.



# The Scenario

- Feature Request: Build a series of dashboards to **report** on our data
  - Blog dashboard for authors
  - Overall dashboard for site admins
- Allow for user-defined filtering
- *Where do you begin?*



# 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 owners
  - What is the total number of blogs by category?
  - How many users are signing up month over month?
  - Which posts are the most viewed?
- For blog authors
  - How many users have viewed a post over the past 7 days?
  - How many comments are left on posts on average?
  - Which are my top 5 rated posts?
  - How many people have viewed posts for category \_\_\_\_\_?
  - How many posts have each of my blog's authors written?

# What is “OLAP”?

- **Online Analytical Processing**
- “...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
    - Buyer
    - Amount
    - Date sold (Date, Month, Quarter, Year)
  - Every combination known and calculated ahead of time
- Large amounts of preprocessed data stored in a warehouse for the express purpose of fast querying
- Commonly deals with aggregate data (count, max, min)

# What is “ROLAP”

- **Relational Online Analytical Processing**
- 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
- Tables organized in a star and/or snowflake schema

# Couple notes...

- We 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

# 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

What is the total number of blogs by category?

How many comments are left on posts on average?

# Dimension

- A point in the data where you can "slice and dice" fact model info
  - Sales rep
  - Date of purchase
  - 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

What is the total number of blogs by category?

How many posts have each of my blog's authors written?



# 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
- Can be part of the metric or user specified

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 total number of blogs by category?

---

---

How many posts have each of my blog's authors written?

---

---

# 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

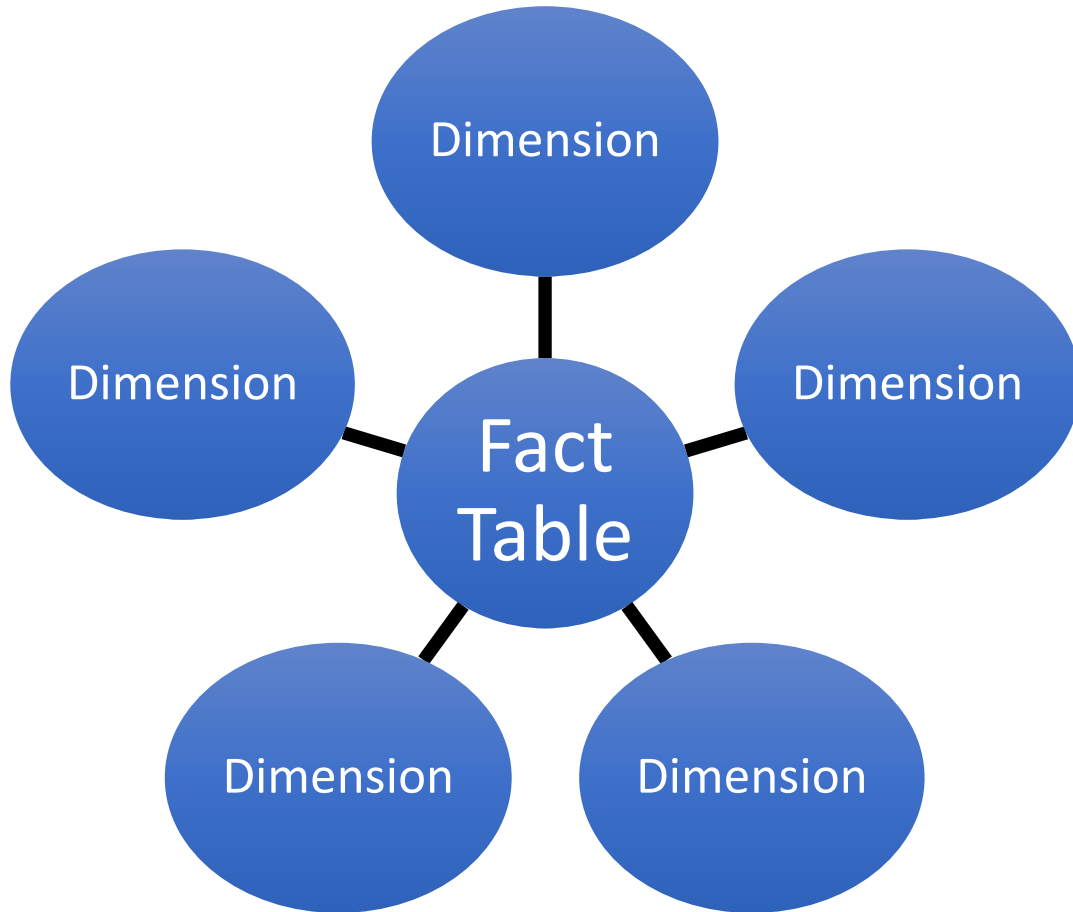
*(comment\_count on posts)*

How many comments were left on posts created today  
grouped by post category?

# Star Schema

- A design pattern for organizing data in a data warehouse
- Allows for queries to be efficient and fast
- Fact table holds core business information / processes
  - A sale
  - A bank transaction
- Fact table holds foreign keys that branch out into dimension tables
- Fact table may also hold non-foreign dimensions as columns
- Other option: Snowflake Schema

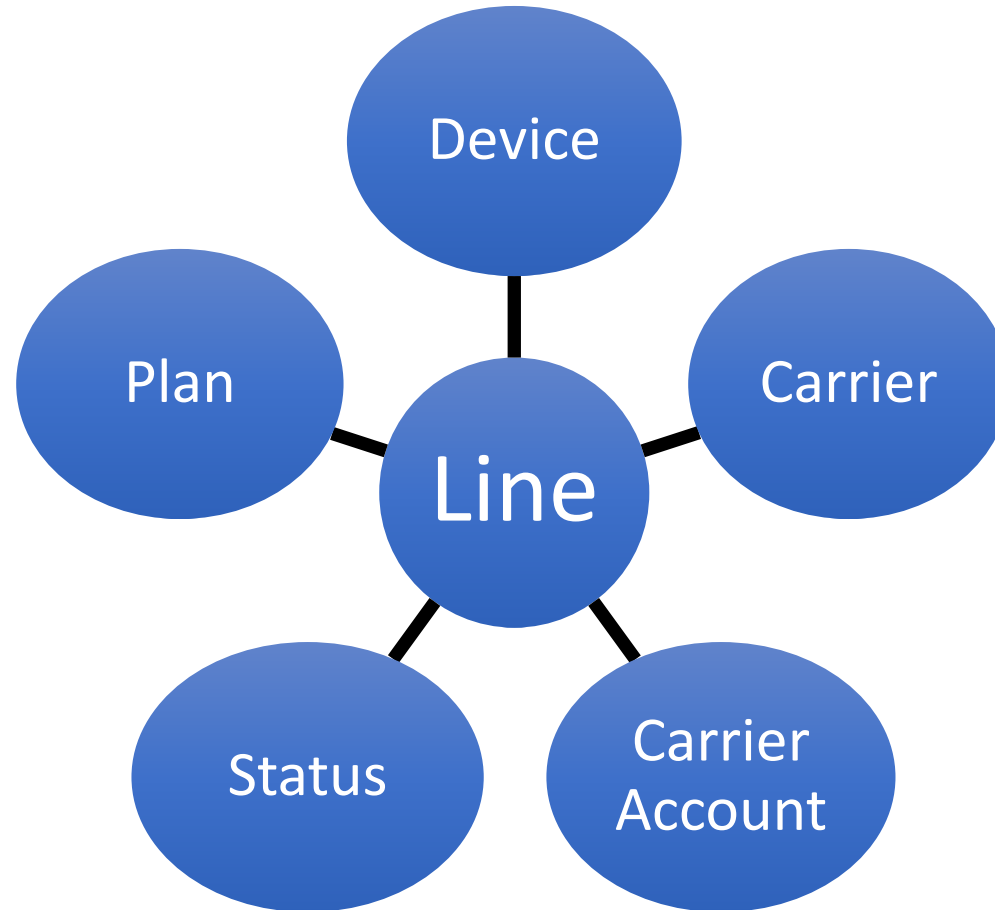
# Star Schema



- Primary driving force of a report in the middle
- Consists primarily of measures, foreign keys and dimensions that live on the fact table
- belongs\_to / has\_one branch out to relations via foreign keys
- More detailed info lives on dimensions
- Prefer dimensions tables over dimensions living on the fact table
- **DOES NOT SUPPORT has\_many relationships (well)**

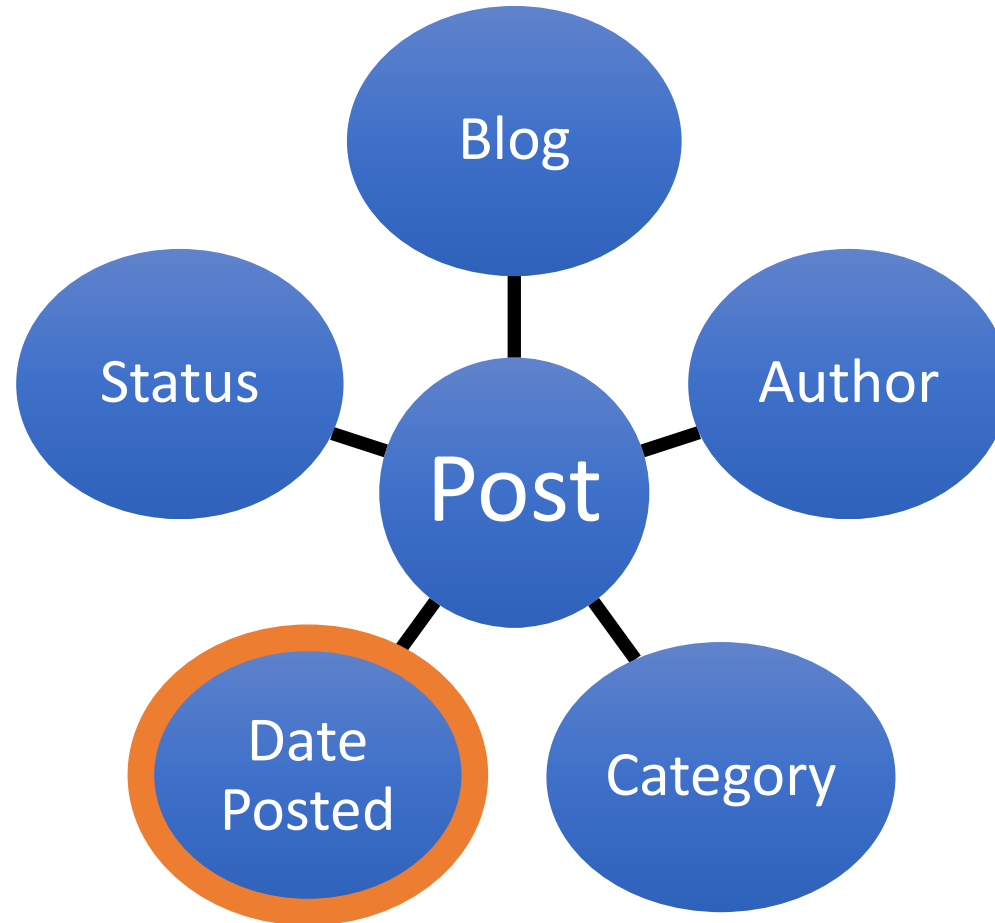


# Star Schema



*Overly simplified example*

# Star Schema

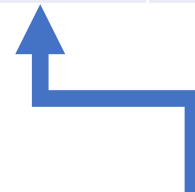


# Dates as Dimensions

- 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



**posts**

id	posted_at_id	blog_id	author_id	title	content
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
  - You can group by relations and get objects back or by columns, not both
  - Can result in multiple queries resulting in ActiveRecord objects instantiated
- Aggregation methods do not allow for full control over multiple columns returned
  - `select()` is ignored
  - `group()` can muck with the `SELECT` clause
- No clear way to have “stock metrics” that can be reusable for different dashboards
- 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 ActiveRecord queries?
  - Not very DRY
  - Complex logic for applying dimensions and filters
- Write a queryer yourself?
  - ActiveRecord provides most of the information you want
  - Could result in dirtying up models
- Switch your application to sequel?
  - Ship's sailed for most apps
  - Requires management buy-in for a rewrite

# active\_reporting

- [https://github.com/t27duck/active\\_reporting](https://github.com/t27duck/active_reporting)
- Implements a DSL for describing fact models, dimensions, filters
- Build ready-made metrics to run reports on
- Uses ActiveRecord to build a query and execute it on the database
- Does not dirty up ActiveRecord (only one new method)
- Not quite done yet (sad face)

# active\_reporting - FactModel

```
class PostFactModel < AR::FM  
  use_model 'Post'
```

```
end
```

```
class Post < AR::Base
```

```
end
```



# active\_reporting – FactModel - Dimensions

```
class PostFactModel < AR::FM
  ...

  dimension :blog
  dimension :category
  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 DateDimFactModel < AR::FM
  ...
  dimension_filter :scope_on_model

  dimension_filter :by_author_id,
    ->(x) { where(author_id: x) }

  dimension_filter :title_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(  
  :post_count_by_author,  
  fact_model: PostFactModel,  
  dimensions: [:author]  
)
```

```
m2 = ActiveReporting::Metric.new(  
  :comment_count,  
  fact_model: PostFactModel,  
  measure: :comment_count,  
  aggregate: :sum  
)
```

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

# active\_reporting – Report

```
metric = ActiveReporting::Metric.new(  
  :comment_count,  
  fact_model: PostFactModel,  
  measure:    :comment_count,  
  aggregate:  :sum  
)
```

```
r = ActiveReporting::Report.new(  
  metric,  
  dimensions: [:blog],  
  dimension_filter: {  
    blog_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

```
> r.run
=> [
  {
    comment_count: 742,
    blog_identifier: 123,
    blog: 'Some Blog'
  },
  {
    comment_count: 623,
    blog_identifier: 456,
    blog: 'Some Other Blog'
  },
]
```

```
SELECT
  SUM(posts.comment_count)
  AS comment_count,
  blogs.id AS blog_identifier,
  blogs.title AS blog
FROM posts
JOIN blogs ON blog.id =
posts.blog_id
WHERE posts.blog_id IN(123, 456)
GROUP BY blogs.id, blogs.title
```

# A Limitation of ActiveRecord

- Extremely difficult to easily join against the same table twice and filter independently

- SupportTicket.  
  joins(:creator, :assignee).  
  where(????)

```
class SupportTicket  
  belongs_to :creator,  
            class_name: 'User'
```

```
  belongs_to :assignee,  
            class_name: 'User'  
end
```

# 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
- Star/Snowflake Schema does not support one-to-many and many-to-many relationships (very well)
- Index liberally where needed
  - Foreign keys
  - Columns commonly used for filtering
  - Use EXPLAIN [ANALYZE]
  - INSERTS will be “slower” as you move to a more warehouse-like database
- Read-only replicating slaves
- Shard or schema separation



# Outgrowing OLTP: Entering “Real” Data Warehousing

- ROLAP functions well enough for OLTP (Transactional) data, but overtime more data leads to slower queries
- Running aggregates on the fly will eventually become too slow
  - Rails counter caches
  - Pre-calculated aggregates (Rebuilt via background jobs)
- Copy/move “historical” data into “rollup tables” and report on them
  - Data is no longer “live”
  - Data is stamped out once
  - Pre-determined aggregates and summation of data
  - Note: Rolling aggregations can cause a loss in dimension data

# The End

- WIP Gem: [https://github.com/t27duck/active\\_reporting](https://github.com/t27duck/active_reporting)
- Slides: <https://github.com/t27duck/showandtell>
- Twitter: @t27duck