tessellation

Creating a reporting data mart.

Six rules and six platforms for optimizing reporting performance.

Overview.

So your IT department set up a data lake a couple of years ago. They promised the world! Well, maybe not the world, but they did promise democratized access to data and increased reporting environment performance. To us analysts, that is as good as promising the world, right?

Guess what?!?! The only delivered on democratization of data. Performance is still a slog, potentially even worse than before! What the hell do you do?

This is one of the most common scenarios we see with our customers. Data lakes are great at storing large amounts of data and giving access to everyone in the enterprise. However, data lakes do basically nothing inherent to speed up reporting performance. Worse yet, it takes a huge amount of knowledge to do anything in the data lake environment to optimize performance. You are reliant on IT to make the changes you need.

So what do you do? Create a reporting data mart of course! How do you do it? Well you have a couple of options, all of which we will discuss.

Before we get into that though, you need to understand the best practices that should be deployed in any reporting data layer. Better yet, your IT department needs to understand the best practices. We have six rules that you need to implement (in any tool or environment) in order to be successful.

6 golden rules for optimizing reporting performance.

The most optimal way of storing data within a database is rarely the best way to optimize performance in our reporting tools. Because of this, we need an in between layer that we will call a ***reporting data mart***.

We will get into our options for reporting data marts in a bit, but first we need to cover some ground rules. When bringing data into our reporting and visualization tools, we must learn the **6 golden rules for optimizing reporting performance**.

Rule #1: distinct measures.

All distinct measures should have their own unique field within the data mart.

Bad structure.

Region	Metric	Value
East	Sales	90
West	Sales	130
North	Sales	80
South	Sales	120
Central	Sales	150
East	Goal	100
West	Goal	125
North	Goal	75
South	Goal	150
Central	Goal	175

We have a problem here. We have a single metric column that contains the labels *sales* and *goal*. We also have a single value column that has the corresponding *sales* and *goal* amount for each of the five *regions.* This is a common data structure that is used to limit the number of columns.

Why is this bad?

The intention here was good. Limiting the number of columns can save storage space and can increase the performance in some scenarios. However, front-end reporting tools expect distinct measures in separate fields. Within our front end tool, we will have to write a calculation to split these metrics into separate columns, which in turn kills performance.

Good structure.

Region	Sales	Goal
East	90	100
West	130	125
North	80	75
South	120	150
Central	150	175

Sales and *Goal* now have their own columns. We now have a structure that requires no calculation on the reporting side. We have also limited the number of rows in the data set, which more often than not is a better way of increase performance, when compared to consolidating columns.

Rule #2: don't partition by dimensions.

All unique measures should be in their own column and should not be partitioned into multiple columns across dimensional values.

Bad structure.

Region	2017 Sales	2018 Sales	2019 Sales
East	70	85	90
West	140	145	130
North	80	90	80
South	115	125	120
Central	75	100	150

We have a problem here, too. We have a single metric *Sales* which is being partitioned into three separate columns, each representing a date field of year.

Why is this bad?

Again, the intention here was good. Limiting the number of rows generally can save storage space and can increase the performance. However, our front-end reporting tools make some assumptions on field structure, just like in our last example. They are expecting unique measures to be in a single field. We will have to write a calculation to coalesce these metrics into a single column, which in turn kills performance.

Good structure.

Region	Year	Sales
East	2017	70
West	2017	140
North	2017	80
South	2017	115
Central	2017	75
East	2018	85
West	2018	145
North	2018	90
South	2018	125
Central	2018	100
East	2019	90
West	2019	130
North	2019	80
South	2019	120
Central	2019	150

Sales and *Year* now have unique columns with their corresponding dimension labels and metric amounts. Yes, we have tripled the amount of rows. This can seem counterintuitive, but if we don't do this ahead of time in our data mart, we would be forcing our front end reporting and visualization tools to coalesce these fields in the same way, which is slow. It will always be quicker to do the preprocessing in the data mart.

Rule #3: metric fields should be additive.

All metric fields should be additive. This means that you can take the sum of the entire column, or a subset of it, and the number returned makes sense.

Bad structure.

Region	Sales
East	90
West	130
North	80
South	120
Central	150
Grand total	570

Take a look at the last row; we have a grand total. Is this actually needed? Let's find out!

Why is this bad?

Front-end reporting tools like to aggregate. If we did a sum of the *Sales* column above, we would have doubled our actual sales amount. Average and max aggregations would also return incorrect results. Like the previous two examples the intention here was good. The data designer thought, "Hey, if they want to see the grand total, they can just filter to the grand total row!" Well, that's the problem. Filters can be slow. Think about it, the system has to find the row that we are looking for. It doesn't know where to find it, even if it is always at the end of our dataset. Aggregation can often be much quicker than a filter.

Good structure.

Region	Sales
East	90
West	130
North	80
South	120
Central	150

Just don't include the grand total or subtotal rows. This is why we removed them in the table to the left.

Rule #4: don't include extra fields.

Don't include extra fields within your data set that are never needed.

Bad structure.

Region	Sales	Unused #1
East	90	100
West	130	125
North	80	75
South	120	150
Central	150	175

We have an extra column here, called *Unused #1*. This is actually the goal data from a previous example. However, in my hypothetical use case, we know we never need this goal data for my reporting.

Why is this bad?

A lot of people want to create a data source that can answer every possible question in the future on a given topic. This is understandable, because it limits our rework of adding more columns in the future. However, there is a big issue. Front-end visualization and reporting tools generally load metadata about every column in a data set, even if that column isn't being actively used. In a situation where we have one extra column, this isn't a big deal. But if we have hundreds of extra columns, it kills performance.

Good structure.

Region	Sales
East	90
West	130
North	80
South	120
Central	150

Pretty straight forward. Don't include columns (fields) that you do not intend on using.

Rule #5: summarize your data.

Limiting the number of rows through summarization is the best way to quickly increase performance in any reporting environment.

Bad structure.

Region	State	City	Sales
East	CT	Hartford	90
East	NY	Albany	130
East	NY	New York City	80
North	WI	Madison	120
North	MN	Saint Paul	150
North	MN	Minneapolis	170

This structure isn't inherently bad. But there might be an issue. Whenever we have a hierarchy, like we do to the right (*Region, State, City*), we need to think critically about what is needed.

Why is this bad?

Any time we have a data set that has a hierarchy, you need to ask yourself - "What is the most granular level of reporting I need to do?"

Let's pretend that we know we only ever want to report at the *overall*, *region*, and *state* levels. In this hypothetical, you know you never want to report at the *city* level. If that is the case, don't include the *city* level. Roll up one level to *state*. This is the easiest way to limit the number of rows, which in turn increases performance.

Please note that Rule #3 about additive metrics applies here as well, at all levels.

Good structure.

Region	State	Sales
East	CT	90
East	NY	210
North	WI	120
North	MN	320

We now only have the rows that we need, eliminating *city* and aggregating the data up to the *state* level.

Rule #6: know where to put your calculations.

Not every metric and KPI we need is going to live in our data warehouse or data lake. That's why we have the ability to make calculations in our business intelligence and analytics tools. But should we be doing that if we can put the calculations in our data mart? Well, the answer is "it depends."

Bad structure.

Region	State	Sales	Profit	Profit Ratio
Central	IL	300	30	10.0%
Central	TX	250	50	20.0%
Central	MI	170	80	47.1%
East	NY	115	15	13.0%
East	CT	165	80	48.5%

Our calculation here is *Profit Ratio*, which is *Profit / Sales*.

Why is this bad?

Having the field *Profit Ratio* in our underlying database is troublesome. At the most granular level, which is the *state* in this case, the profit ratios make sense. However, if we want to report at a higher level for *profit ratio*, such as *region* or the overall ratio, we couldn't use the existing *profit ratio* field. This is because the field breaks rule #3; it isn't additive. We can't sum it or average it to get the actual profit ratio for the higher levels in the hierarchy. Because of this, we should not have this type of calculation in our data set and we should rely on our reporting tool to calculate it.

That said, it is always better for performance to have the calculation in the underlying data, but there are only a certain type of calculation that should be included.

Good structure.

Region	State	Sales	Profit	Cost
Central	IL	300	30	270
Central	TX	250	50	200
Central	MI	170	80	90
East	NY	115	15	100
East	CT	165	80	185

We have a new calculation here: *cost.* This is a much better calculation to have in our underlying data. It doesn't break rule #3 and is additive.

6 rule summary.

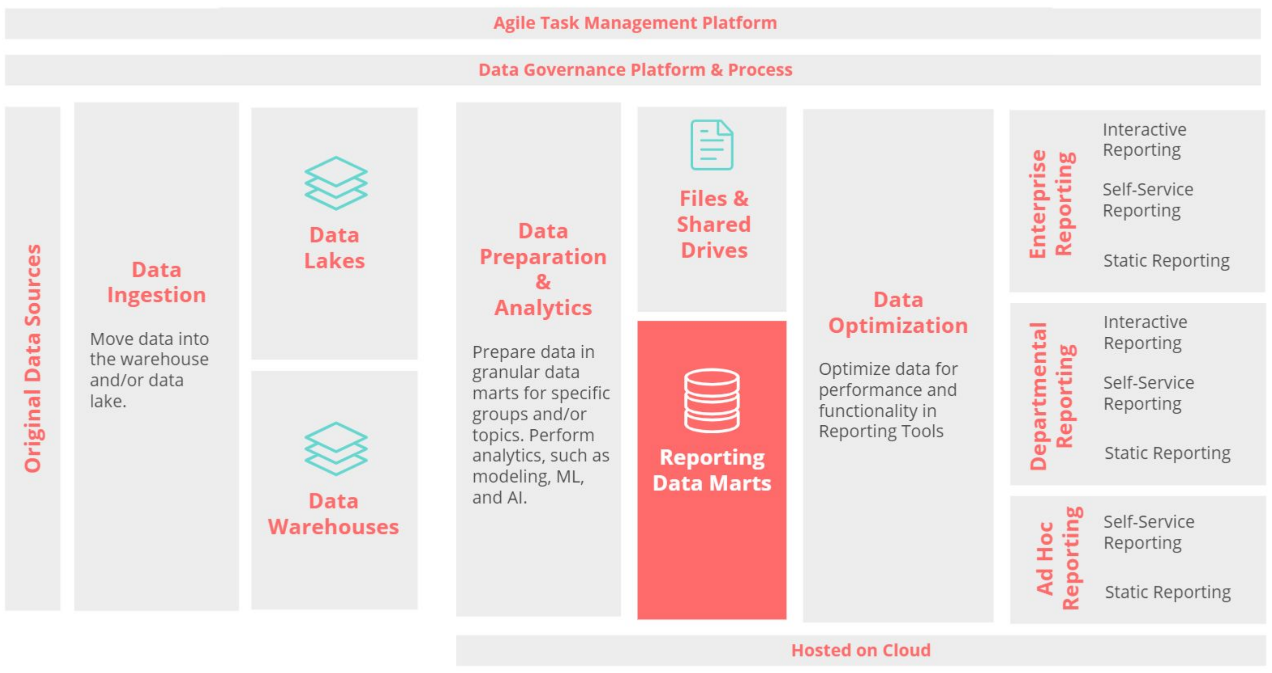
These six rules are a great starting place to understand how to get the most out of your data for reporting and analytics. However, these are generalities. There will always be exceptions to these rules.

The case for reporting data marts.

What is a reporting data mart?

So, what is a reporting data mart? This term often refers to a (generally) self-service that enables users to grab the data they need for analysis and reporting.

The reporting data mart is a data layer that sits between the underlying data warehouses / lakes and the front-end reporting environment.



This contrasts with data warehouses and data lakes. The tables within a reporting data mart tend to be significantly more curated, optimized for performance, and are often organized by topic. Nowadays, data marts are generally easy-to-use web-based platforms. They need to be as they are (generally) serving an audience that is less technical.

Why do we need a reporting data mart?

We have now learned how to structure data before reporting and analytics, but why do we need the data mart layer? Why can't we just apply these rules in the underlying data warehouse or data lake?

The same rules don't apply in the data lake.

The **6 golden rules** for a data mart don't necessarily apply to a data warehouse or data lake. The ingestion of data and the optimal storage in these data systems is often at odds with the rules that we went through. This is why a new layer, the reporting data mart, is needed.

Ease of access.

Most data mart technologies are easier to use when compared with data lake technologies. They are designed with business and analysts in mind. They don't require the same expertise to understand as you would see in a purely IT-led tool.

Keeping it topical.

Data marts are generally topical, meaning they are targeted for a specific use case (or small set of use cases). This inherently means that data sets are smaller, in number of rows and columns, which in turn greatly improves performance. Also, this means that it is much easier to find what you are looking for as the environment is highly curated.

Applying the 6 rules in a reporting data mart.

There are scores of different ways to apply our learnings about creating a reporting data mart. We have six common options where we have seen success.

Within the data lake.

You don't have to deploy a new technology to create a data mart. You can deploy it within your existing data lake by having IT create a new reporting-certified layer. Our recommendation is Cloudera Hadoop.

Advantages.	Disadvantages.
Infrastructure already exists.	Slow to create and deploy.
Cost effective.	No additional performance optimization.
Access to all of your data.	Not user friendly for business.
Scales well.	Slows existing infrastructure.

Within data visualization tool.

Most data visualization tools allow for the creation of targeted data extracts. This is essentially a file mart that can be used in place of server-based reporting data mart. Our recommendation is Tableau.

Advantages.	**Disadvantages.**
Infrastructure already exists.	Proliferates file-based data sources.
Cost effective.	Slows existing infrastructure.
User friendly for the business.	No additional performance optimization.
Quick to create and deploy.	Extremely limited data prep functionality.
Centralized in one business tool.	Very manual. Doesn't scale well.

Within data preparation tool.

The market now has many data blending and preparation tools that makes it easy to create a data mart, whether it be to a file mart or data server. Our recommendation is Alteryx.

Advantages.	**Disadvantages.**
User friendly for the business.	Proliferates file-based data sources.
Flexible.	No additional performance optimization.
Large amount of data prep functionality.	Very manual. Doesn't scale well.
Quick to create and deploy.	Moderate cost.
Additional analytics capabilities.	Security concerns.

Within the data lake optimization tool.

There are many data lake optimization tools that essentially create a data mart on the fly, based on usage patterns and virtualization. Our recommendation is AtScale.

Advantages.	Disadvantages.
User friendly for the business.	No additional performance optimization.
Scales well.	Slow to deploy.
	High cost.

On-premise SQL server.

You can use a dedicated SQL server as a reporting data mart. Our recommendation is Microsoft SQL Server 2017.

Advantages.	Disadvantages.
Additional performance optimizations.	Not user friendly for the business.
Additional analytics capabilities.	High cost (software and hardware).
Secure.	

Cloud-based data mart.

Cloud-based data marts are the most flexible way to deploy a performant reporting data layer. Our recommendation is Snowflake.

Advantages.	**Disadvantages.**
Easy deployment for existing cloud companies.	Difficult deployment for existing cloud companies.
Cost scales based on usage and storage.	Potentially difficult integrations to on premise technologies.
Is actually secure.	Perception that cloud is not secure.
Additional cloud integrations for analytics.	
Additional performance optimizations.	
Scales well.	
User-friendly for the business.	
Quick to create and deploy.	

Final recommendations.

All data-driven companies, especially those prioritizing self-service, should create a reporting data layer. Following our six golden rules will help optimize this data layer.

Any of our six recommendations for reporting data mart deployments are valid, but they are not created equal. Tessellation strongly feels that cloud-based data marts are the most flexible way to deploy a performant reporting data layer. They are user-friendly, quick to deploy, and integrate well with other cloud technologies. Our overall recommendation for reporting and analytics data marts is Snowflake.

About taessellation.

What we do.

We partner with organizations to help them make better decisions. And we don't mean this in a superficial way. We are coaching executives. We are training analytics professionals and their stakeholders how to think with data and bring about fact-based change. And when organizations don't have the talent they want or need to do the work that is required for change, we directly manage and support analytical tool development. We're even building our own go-to-market tools for our customers in CPG, retail, and healthcare.

Who we are.

We are a team deeply connected to our respective analytics communities. We are constantly sharpening our craft by collaborating at (and outside of) work. Our team is obsessive over doing things the right way and making a true impact with our clients. We are preoccupied over the quality of our product. Our team is authentic, curious, and intellectually humble. We have fun, make extra effort to build and celebrate a diverse workforce, and have an amazing work family.

Why we are different.

The technological landscape is extremely different than it was just three years ago. Large on-shore consulting firms are shifting their focus to AI, ignoring daily analytics decisions. These decisions are left in the hands of often underskilled analysts. They assume what they are seeing in tech-forward cities is permeating throughout the rest of the country and world. It's just not. And while AI is experiencing a renaissance, organizations can't truly move forward until their rank-and-file have adept analytical skills. That's where we are committed.

There are some consulting companies still invested in analytics, but they lack quality throughout their organizations to meaningfully support clients. We hear one story over-and-over: an organization hires an on-shore/off-shore consulting firm with name recognition to support the development of their backlog. The products returned are low quality, take too long to develop, and don't perform well. And while the reports are exactly what they are asking for, they oddly are not helpful. This is where having high-caliber talent can make a huge difference: we develop higher quality, performant solutions, and we do it faster. We ask questions and understand the needs of our clients and develop analytic products that go beyond initial requirements.

About the author.

Alex Christensen is a partner and co-founder at Tessellation. He oversees technology, engineering, and partnerships for the company.

Alex is passionate about enabling end-to-end analytics environments and has focused his career on growing and sustaining analytics centers-of-excellence. He is a "jack-of-all-trades", with a near-encyclopedic knowledge of the entire analytics technology stack.

Alex is motivated by empowering clients. He is a coach at heart and loves to show clients how they can be more self-sufficient and successful.

Alex lives in Saint Paul, Minnesota.

Alex Christensen | alex.christensen@tessellationconsulting.com | (715) 207-8877