The Details: Data Warehouses

The post you have all been waiting for

Justin
Mar 29, 2021

15 3

Hello there, lovely paid subscribers. This week's post is going to dive deeper into one of Technically's most requested topics: **data warehouses**. We're going to look at what they're used for, popular options (Snowflake, BigQuery, Redshift: what's the difference?), how data lakes fit into the picture, and how vendors are positioning themselves as *platforms* going forward.

The first "Details" post covered ETL.

Refresher: what's a Data Warehouse?

A data warehouse is a specially designed database that holds **analytical data**. It's built to handle long, complicated queries written by data scientists, analysts, and machine learning engineers.

If you're iffy on what data warehouses do, now would be a good time to read the original post here. It covers:

- Why we use separate databases for our **apps vs. analytics**
- How data gets into a warehouse via **ETL or ELT**
- Typical types of **analytical queries**

And then come back here.

> 🚨 **Confusion Alert** 🚨
>
> Part of where data warehouses get confusing is **what** they actually are. And the answer is that a data warehouse involves both:
>
> 1. A different **place** to store your data (a separate database server)
> 2. A different **way** of storing data (new data sources, structures, etc.)
>
> Most companies with data teams will have at least two separate places where they work with data: their production database, and a data warehouse. The production database only has data that's relevant to the core operations of their app (users, business concepts, etc.). The data warehouse, on the other hand, might have copies of their production data, payment data, website traffic data, and lots of other stuff - whatever the data team wants to analyze.
>
> 🚨 **Confusion Alert** 🚨

All good? Cool – now let's dive into the details.

Different types of warehouses, home rolled to managed

There are basically 3 levels of data warehouse – and companies can be at any of these levels. The first two options are **home rolled** – meaning the company pays for and maintains their own infrastructure – while the third option is a managed service, and outsources infrastructure to someone like AWS.

1. **Home rolled: basic SQL**

The most basic type of data warehouse is just a vanilla setup of a typical SQL database like Postgres or MySQL. It might be sitting on a bigger server than your production database so it can handle more complex queries, but under the hood it's the same technology.

What makes this a data warehouse, instead of just a regular database, is that (a) it's more powerful if you put in on bigger servers, and (b) it has analytical data in it instead of just production app data.

> 🛠 **Workplace Example** 🛠
>
> At one company I worked at, we had Segment automatically pipe out event

> data into Postgres tables, and queried them as our "warehouse" – even though it was just a basic Postgres database.
>
> 🙏 **Workplace Example** 🙏

This setup is not very popular these days. An even rarer combination would be using a NoSQL database like MongoDB or DynamoDB.

2. **Home rolled: big data**

Option number 2 is setting up your own infrastructure and using more specialized, **analytics-focused open source** like Hadoop. This is what we did at DigitalOcean before moving onto Snowflake (see section #3):

- Our data was stored in HDFS (Hadoop), set up and running on our own servers
- We used Apache Hive to query it using a SQL-like interface
- Our queries ran on Presto, which distributed them across powerful servers

This was a decently complicated setup, but not uncommon. The benefit to rolling your own systems tends to be cost – we didn't pay per query, and were able to store and query a *lot* of data. But the downside was maintenance: setting up and maintaining the warehouse was *tough work*, and there was constant downtime.

And that's why we eventually migrated to Snowflake!

3. **Managed services: Snowflake, BigQuery, Redshift**

Finally, we arrive at what's all the rage these days: **cloud native, fully managed data warehouses**. When you buy something like Snowflake, they take complete care of infrastructure, and you don't need to manage any servers whatsoever. The basic highlights:

- Fully managed infrastructure (don't need to touch servers, automatic scale, etc.)
- Pay per use model: usually $/storage and amount of data queried
- Slick browser-based user interfaces for managing permissions and data

These are completely taking over the market – I mean for fuck's sake, Snowflake was the largest tech IPO literally ever.

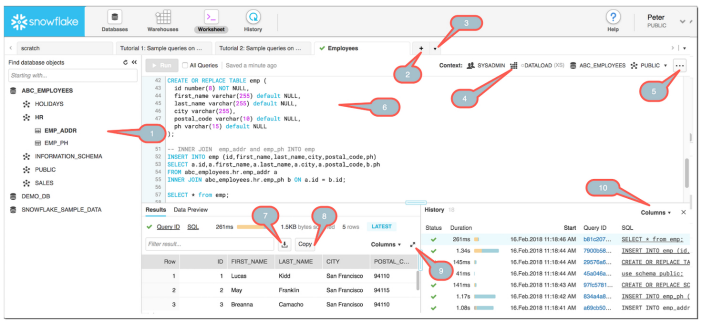
One of Snowflake's major innovations was the **separation of storage and compute**. Instead of sticking data warehouses on giant servers with powerful CPUs *and* a lot of attached storage, teams started to separate the two – data stored on network-attached storage for cheap, and big servers for running expensive queries that can scale up or down easily. And this separation also made its way into pricing, where you now get charged for storage and compute (queries) separately.

Snowflake, BigQuery, Redshift, oh my!

Snowflake has definitely made the biggest splash in the data warehouse space, but they are not the *only* legit option. This section will run through some of the popular vendors when you opt for a managed service. For a more in depth look, check out this lovely comparison post.

→ Snowflake

Snowflake probably has the lead in terms of enterprise adoption (according to my very scientific conjecture). Here's what the basic UI looks like:



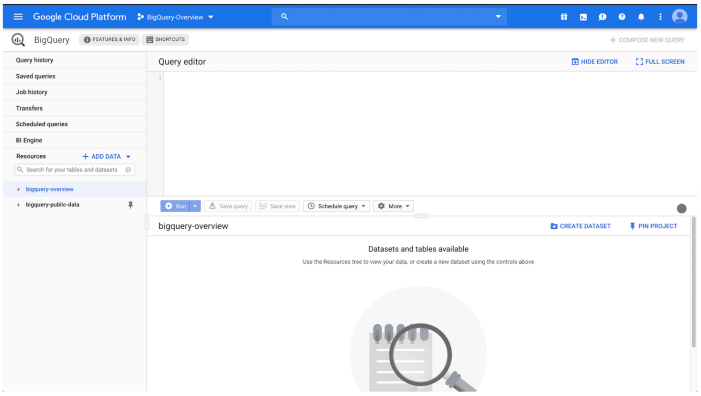
Snowflake highlights (i.e. what make it unique) include:

- Support for diverse file types (Parquet, Avro, other random names)
- High quality permissioning and data sharing capabilities
- Credits-based pricing model, charging for query run time

Part of what makes Snowflake attractive to larger enterprises is the fact that it is, in theory at least, **multi-cloud**, while BigQuery (Google) and Redshift (AWS) lock you down to one cloud provider. You can run Snowflake on all 3 major clouds – Amazon, Google, and Microsoft. Now whether people actually *use* that multi-cloud capability, or just want to be able to tell their boss that they *could* use it, is a topic for another time.

→ BigQuery

BigQuery is Google Cloud's data warehouse, and the one I have the most direct experience with (i.e. a lot). Here's what the basic web interface looks like:



BigQuery highlights:

- Really nice integration with other GCP products (event streams, functions)
- Pricing based on query sizes (not runtime) and storage

The whole Snowflake vs. BigQuery discussion is hard to have, since they're really so close in terms of features and performance. You can read about speed and performance benchmarks, but either one can look better in different use cases / situations. Again, my hunch is that most of the decision tends to come down to whether organizations care about multi-cloud or not, which is a longer discussion for another time (read: never). Also: startups tend to make their decision based on which cloud provider gave them the most startup credits.

→ Redshift

Redshift is tough. On the one hand, it's usually in the discussion with Snowflake and BigQuery, but it's kind of hard to figure out why – by most standards it's not nearly as good as either. Redshift doesn't separate storage from compute, can't really auto-scale, and tends to lose out on performance benchmarks.

So it's worth mentioning, but companies I've worked at haven't really seriously considered it as an option. One thing that is worth noting is Firebolt, a new entrant built on top of AWS that claims to be the fastest option for certain classes of queries. Keep an eye out for them.

What's a Data Lake, then?

Because those in the data profession seem to be obsessed with naming things after water-related concepts, you may have heard of a data lake. What's that, and how does it relate to a warehouse?

Remember that data warehouses are optimized for **analytical** queries, often in real time. They're made to be used, constantly. And if you're paying for something like Snowflake or BigQuery, they come with a nice web UI for writing queries and managing permissions. Data warehouses are **not the cheapest way** to store data, because they're optimized for **using and querying** that data.

Data lakes, on the other hand, are optimized for **long term storage**. You can think of them as like a storage unit for your data, where you just throw stuff in there - there's no good way to query them, and you probably want to take your data out first before you work with it. What makes a data lake a data lake:

- Data is stored in its **native format** (be it an ugly, unusable file)

- **Storage is really, really cheap** compared to something like Snowflake

Companies with very specific, scaled data storage needs (think: internet of things sensors, tons of text data, etc.) will usually have data lakes as part of their tech stack. As the need arises – say a team is building an ML model – they'll pull the data out and use it somewhere else.

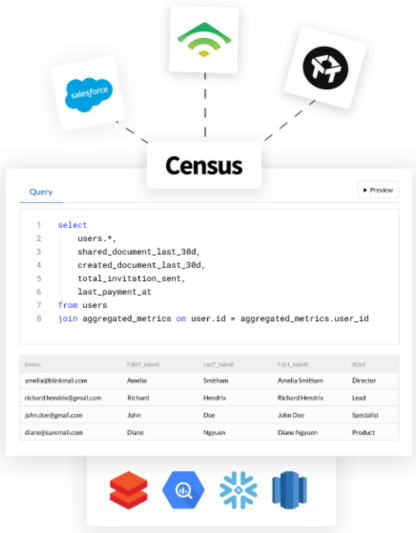
> 🔍 **Deeper Look** 🔍
>
> When looking at data lakes, you'll see the "schema defined when" terminology used a lot. When you store data in a warehouse, you need to define your schema (tables, column names and types, relations, etc.) up front. In a data lake though, you kind of just shove the data in there, and define the schema when you get the data *out*.
>
> 🔍 **Deeper Look** 🔍

This is where things get interesting, though – there's a (fringe?) movement towards actually **querying the data lake directly**, or in other words, blurring the lines between cheap storage and useful queries. The OG here is AWS's Athena, which lets you query your raw data in S3 (popularly used as a data lake) and define your schema on the fly. The open source Delta Lake project (which seems to be bankrolled by Databricks, low key) is trying to bring transactional rigidity to data lakes, too. TL;DR: keep an eye out.

The data warehouse as a platform

The word "platform" is obviously trite at this point, but it's worth thinking about where companies like Snowflake go from here. And their explicit product direction is towards **the warehouse as a platform** – or in normal people words, helping other companies build products and services on top of the data warehouse.

The best example of the "warehouse as a platform" concept that comes to mind is this new buzzword going around, **Reverse ETL**. I'll write an issue on this eventually, but in short, it means the process of getting your valuable warehouse data **out of** the warehouse and **into** your operational tools, like Salesforce for your sales team or Hubspot for your marketing team. Companies like Census and Hightouch offer this as an easy-to-use SaaS tool.



In other words, it's the warehouse providing value beyond just querying it, hence the "platform" moniker.

The balance for Snowflake is in offering the right granularity of APIs to let people like Census build dope services on top of them, vs. building out these features themselves so they get all the economic upside. I'd expect this tension to exist across the whole warehouse stack, like:

- Notebooks and feature stores for data science
- Modeling and ETL/ELT tools (e.g. dbt)
- Data lineage and discovery tools (e.g. SelectStar)

If you have any thoughts or comments, just jump in below!

♡ 15 | 3 | Share

Write a comment...

Brandon Apr 16, 2021

Would love for you to actually do the post on whether or not organizations really care about multi-cloud!

Reply

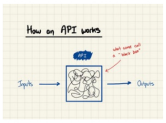
Gary Schnierow Mar 31, 2021

Agree, a great post, wondering about MongoDB too. Also wondering about Tableau and other business intelligence and analytics and if that is complementary needing Snowflake's data warehouse and ability to query or competitive (fearful of you saying both)? Thanks!

Reply

1 more comments...

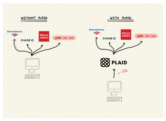
Top | New | Community



What's an API?

What McDonalds and Lyft have in common

Justin Jan 9, 2020 ♡ 187



What does Plaid do?

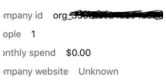
Technically begrudgingly tackles Fintech

Justin Jan 14, 2021 ♡ 34 3

What does New Relic do?

Keeping an eye on your servers and apps

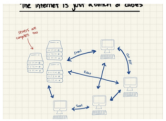
Justin Jan 11 ♡ 23 2



What's Reverse ETL?

Getting your data OUT of your warehouse?

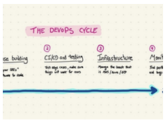
Justin Feb 1 ♡ 19



What happened to Facebook?

A basic explainer of what that outage was all about

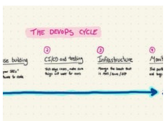
Justin Oct 5, 2021 ♡ 29 4



What does GitLab do?

The TL;DR GitLab is a somewhat contrarian take on DevOps: it's basically one giant tool for literally anything you'd want to do relating to building and...

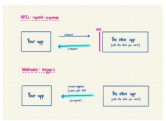
Justin Jan 4 ♡ 8



What's DevOps?

IT has a cool new name

Justin Jan 5, 2021 ♡ 34



What are webhooks?

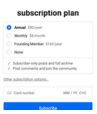
Triggered

Justin Sep 13, 2021 ♡ 28 5

What's Headless E-Commerce?

We may be running out of names

Justin Nov 2, 2021 ♡ 20 7



I built a (basic) Substack clone in a month

This was probably a waste of my time

Justin Sep 2, 2020 ♡ 50 4

See all ›

© 2022 Justin · Privacy · Terms · Collection notice

Publish on Substack

Substack is the home for great writing

Our use of cookies
We use necessary cookies to make our site work. We also set performance and functionality cookies that help us make improvements by measuring traffic on our site. For more detailed information about the cookies we use, please see our privacy policy.