Re-Imagine your Data Pipelines Using dbt

Honish Joseph, Derik Roby

Abstract: ***dbt [1] has quickly become one of the most popular tools for managing transformations in ELT data pipelines, offering a fresh approach to data engineering workflows. By borrowing tried - and - tested practices from software engineering, dbt brings much - needed consistency to workflows that often feel chaotic or disconnected. It simplifies pipeline development, takes care of dependency management automatically, and builds testing and documentation directly into the process. This article explores how dbt empowers teams to build scalable, high - quality data pipelines, improve collaboration, and reduce dependency on specific platforms, making data projects more efficient and future - proof.***

Keywords: dbt, data pipelines, ELT, modern data stack, Data Engineering, Data Warehouse

1. Problem

Building a conventional data pipeline in ETL or ELT involves using various tools. In traditional ETL pipelines, which are still common in large enterprises, dedicated ETL tools like SSIS and Informatica connect to source systems, transform data, and load it into data warehouses. These pipelines, often referred to as 'jobs' before the rise of data pipelines, require careful sequencing based on dependencies. Achieving this requires architectural expertise in the project's data flow and job prerequisites. If extra steps like data testing are needed, additional tools are integrated. However, building and managing these pipelines are challenging because they demand specialities in multiple tools. All these tools must work together to complete batch runs, making data pipelines a bundle of tools that are difficult to create and maintain.

Modern ELT teams rely on data integration scripts/tools, high - performance databases, and languages like SQL, R, and Python, along with powerful visualization tools. Despite improvements, integrating tools remains a challenge in modern ELT data pipelines. Even with these modern tools, Data Engineering (DE) and Analytics projects often struggle to deliver quality, responsive data pipelines and analytics [4]. This often results in low or even negative Return on Investment (ROI).

One primary reason for these challenges is the absence of a proper workflow in DE/Analytics projects. A workflow for development includes version control, quality assurance (QA), dedicated environments, documentation, and modular code. It provides a structure for managing changes, ensuring data quality, and facilitating code understanding. In its absence, analysts may redo work and often experience difficulties in understanding unfamiliar datasets.

2. What is dbt?

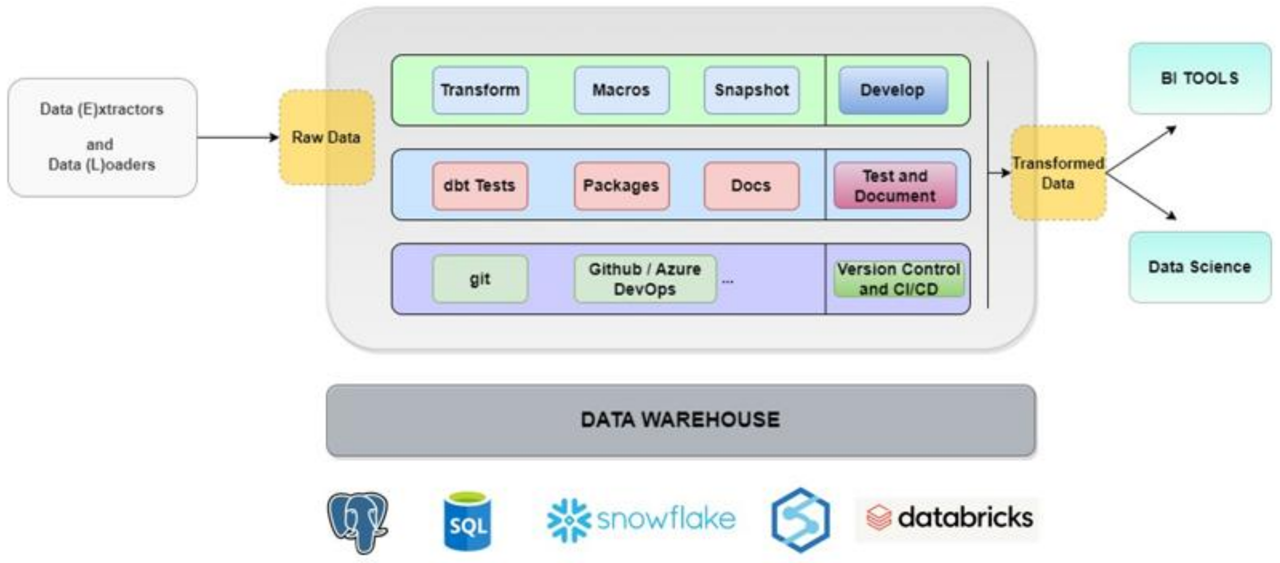

Figure 1: dbt in modern data stack

dbt is at the core of the modern data stack, serving as the transformation tool in EL (T) data pipelines. It's a free, open - source Python tool that you can install and run on any system. Alternatively, you can use dbt - cloud for a fee. With dbt, you write models (transformations) using SQL and a bit of Jinja. Then, dbt compiles this code into SQL that works directly with your data warehouse—whether it's Databricks, Snowflake, SQL Server, or another supported system. This ensures that your data remains secure within the warehouse.

Once you've written the transformations, dbt can automatically determine the dependencies between these models (objects in the database) and execute them in the right sequence. This eliminates the need for separate orchestration tools and the need for personal expertise in the architecture. dbt also offers built - in data testing capabilities for these models. This means you can embed data quality checks into your pipeline without the need for additional tools.

dbt supports documentation as well, enabling data engineers and analysts to document their models without relying on extra tools. This documentation even includes a data catalog with features like a lineage graph, which fulfils the documentation needs of many data teams. dbt is backed by a growing community that constantly improves it with each iteration.

How dbt addresses the problem?

- **dbt helps in the unbundling of data pipelines**

dbt simplifies data pipeline management. After writing SQL transformations in dbt, the tool automatically identifies dependencies between various transformations and constructs a Directed Acyclic Graph (DAG) for the pipeline run. This eliminates the necessity for separate orchestration tools and relieves Data Engineers and analysts of the painful task of dependency management. With built - in testing and documentation features, as well as the ability to script custom tests, dbt eliminates the need for unnecessary testing or catalog tools in your data stack.

- **dbt helps to simplify your team’s technical stack**

With dbt, data pipelines can now be built using SQL and Jinja. It doesn’t require the specialized tooling expertise often demanded by traditional ETL/ELT data projects. This enables organizations to empower their analytics teams to create data pipelines that fulfil their analytical needs, reducing the workload on data engineers. Data engineers can then concentrate on improving upstream data accessibility or addressing other critical tasks that demand specialized skills. For instance, in data pipelines, dbt automatically handles the entire DDL (Data Definition Language) without the need to explicitly create or manage it. This concept of using the analytics team to create data pipelines, has gained traction, particularly among the teams that work with dbt. However, it’s important to note that the analytics team should possess proficiency in data warehousing and architectural design.

- **dbt provides a workflow for DE / Analysts [2]**

dbt draws its workflow principles from well - established practices in software engineering. With dbt, projects can be tracked and version - controlled using Git. It also facilitates the implementation of a CI/CD (Continuous Integration/Continuous Deployment) workflow in data projects. To ensure the production of high - quality data pipelines, strict quality assurance checks and data quality tests can be easily implemented. Additionally, coding style standardization is achievable through SQL linting tools, allowing the DE and analytics teams to dedicate more time to higher - quality work. One notable feature of dbt is its out - of - the - box support for dedicated environments for developing data pipelines, such as development (dev), testing (test), and production (prod) environments.

- **dbt encourages the implementation of software engineering best practices in data engineering workflows [3]**

dbt provides a structured framework for creating modular transformations and tests that can be easily reused across your dbt project. With dbt best practices, dbt developers can ensure the maximum portability of their code, avoiding dependencies on specific target systems. As a result, your data project’s business logic (which is stored and version - controlled as code) is capable of running on various target systems, such as Databricks or Snowflake, with minimal migration efforts. By implementing a well - defined workflow, you can ensure that your codebase scales nicely with growing complexity and team size, preventing it from exploding. This approach enables data projects to function efficiently, much like software projects, where multiple team members can collaborate seamlessly.

3. Conclusions

There is no doubt that dbt has sparked a revolution in the modern data stack. It has transformed the way data engineers build data pipelines. By leveraging established software engineering practices, dbt improves data engineering workflows, simplifies development, and ensures better quality outcomes for data projects. However, like any tool, dbt can lead to issues without proper guidelines in place. This emphasizes the importance of data engineers and analysts having a solid understanding of dbt best practices [2].

Another significant advantage is dbt’s ability to create data projects with minimal dependency on specific target systems, reducing the risk of vendor lock - in for organizations. Since dbt projects are built using SQL, they are likely to be around for a much longer time, as per Lindy’s effect (the older something is, the more likely it will remain relevant in the future). Therefore, dbt is less likely to become outdated quickly, unlike many data tools of its time.

References

[1] dbt. (2024). *Dbt: Data build tool*. https: //www.getdbt. com/
[2] dbt Labs. (2024). *Dbt best practices*. https: //docs. getdbt. com/guides/best - practices
[3] Pradip Sodha. (2024). *Dbt and software engineering*. https: //dev. to/sudo_pradip/dbt - and - software - engineering - 4006
[4] Staff, D. (2020). *Most data science projects fail – but yours doesn’t have to*. https: //www.datanami. com/2020/10/01/most - data - science - projects - fail - but - yours - doesnt - have - to/