

# Implementing Data Versioning & Testing in dbt

by Fariz Wakan

April 24th, 2024



# Data versioning with dbt

## Dimensional modeling

- Dimensional modeling is a **data modeling technique** where you break data up into “**facts**” and “**dimensions**” to organize and describe entities within your data warehouse
- The purpose of dimensional modeling is to **optimize the database for faster retrieval of data**

## Dimensional model vs Relational model

- A dimensional model is **designed to read, summarize, analyze numeric information** like values, balances, counts, weights, etc. in a data warehouse. In contrast, relational model is **optimized for addition, updating and deletion of data** in an OLTP system.
- In the relational model, **normalization and ER models reduce redundancy in data**. On the contrary, dimensional model arranges data in such a way that it is easier to retrieve information and generate reports, ***denormalized and redundant***.

# Facts & Dimensions

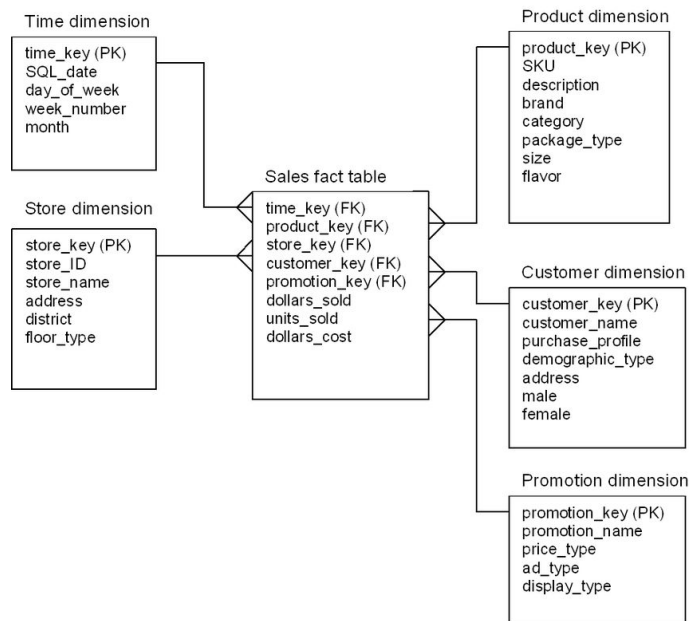
- **Facts**

The **measurements/metrics** or facts of the business process

- **Dimensions**

The **context** surrounding a business process event. It give who, what, where of a fact.

# Fact & Dimension Tables

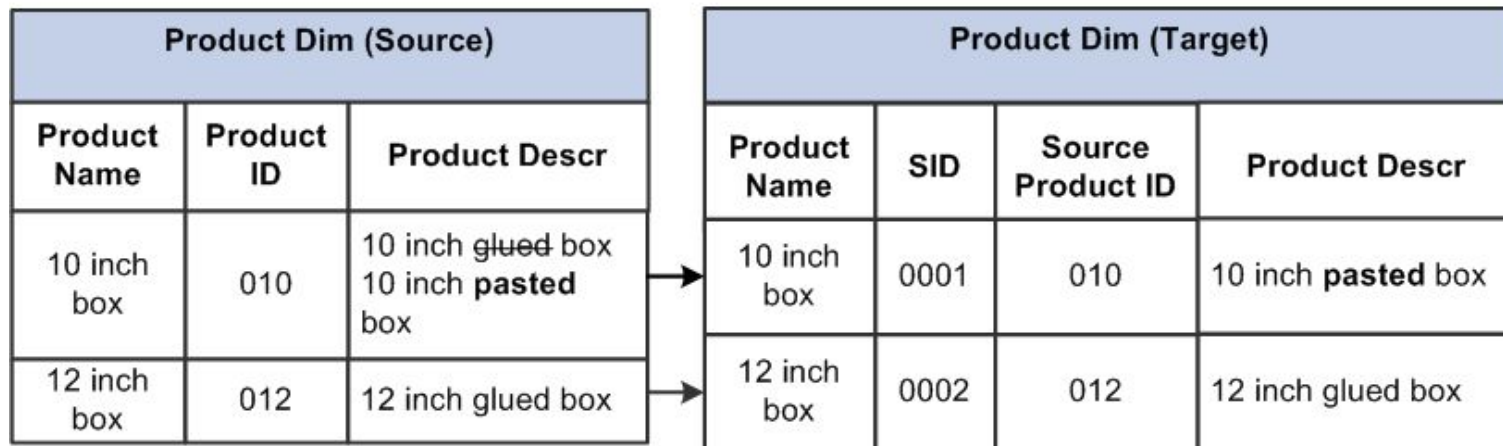


## Slowly Changing Dimension (SCD)

- SCD is a dimension that **stores and manages both current and historical data over time** in a data warehouse
- The three types of SCDs,
  - Type 1 SCDs - *Overwriting*
  - Type 2 SCDs - *Creating another dimension record*
  - Type 3 SCDs - *Creating a current value field*

# Type 1 SCD

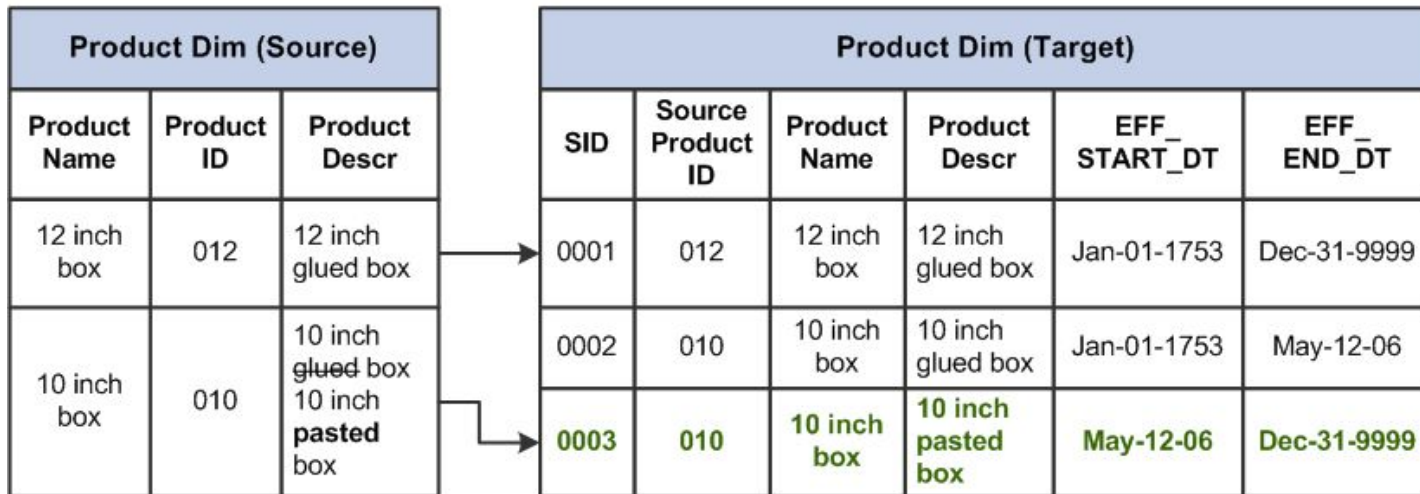
## Type 1 Slowly Changing Dimension





## Type 2 SCD

### Type 2 Slowly Changing Dimension



## Type 3 SCD

cust_id	customer_number	first_name_pre	fist_name_curr	last_name	effective_date
1	10001	NULL	Kontext	Wonderful	2022-01-01



SCD Type 3 Merge

cust_id	customer_number	first_name_pre	fist_name_curr	last_name	effective_date
1	10001	Kontext	Context	Wonderful	2022-07-01

# dbt supports SCD ..

The screenshot shows the dbt documentation website. The main heading is 'Add snapshots to your DAG'. Below it, there's a section for 'Related documentation' with links to 'Snapshot configurations', 'Snapshot properties', and 'snapshot command'. A section titled 'What are snapshots?' explains that analysts often need to 'look back in time' at previous data states in mutable tables, and dbt provides a mechanism for this through snapshots. It mentions that snapshots implement 'type-2 Slowly Changing Dimensions' over mutable source tables. An example is given of an 'orders' table with a 'status' field that can be overwritten as the order is processed. At the bottom, a table snippet is visible with columns 'id', 'status', and 'updated\_at', and one row with values '1', 'pending', and '2019-01-01'.

dbt

Docs Reference v1.7

Resources Community Search Create a free account

What is dbt?

Guides

Supported data platforms

About dbt Cloud

Set up dbt

Develop with dbt Cloud

Build dbt projects

About dbt projects

dbt tips and tricks

Build your DAG

Models

Tests

Snapshots

Seeds

Home > Build dbt projects > Build your DAG > Snapshots

## Add snapshots to your DAG

### Related documentation

- Snapshot configurations
- Snapshot properties
- snapshot command

### What are snapshots?

Analysts often need to "look back in time" at previous data states in their mutable tables. While some source data systems are built in a way that makes accessing historical data possible, this is not always the case. dbt provides a mechanism, **snapshots**, which records changes to a mutable table over time.

Snapshots implement type-2 Slowly Changing Dimensions over mutable source tables. These Slowly Changing Dimensions (or SCDs) identify how a row in a table changes over time. Imagine you have an `orders` table where the `status` field can be overwritten as the order is processed.

id	status	updated_at
1	pending	2019-01-01

### CONTENTS

Related documentation

- What are snapshots?
- Example
- Detecting row changes
- Timestamp strategy (recommended)
- Check strategy
- Hard deletes (opt-in)
- Configuring snapshots
  - Snapshot configurations
  - Configuration best practices
- Snapshot query best practices
- Snapshot meta-fields
- FAQs

Edit this page

# dbt snapshot

Demo Session



# Data testing with dbt

...

The screenshot shows a web browser displaying the dbt documentation page titled "Add data tests to your DAG". The URL in the address bar is `docs.getdbt.com/docs/build/data-tests`. The page has a dark theme with a sidebar on the left containing navigation links like "What is dbt?", "Guides", "Supported data platforms", "About dbt Cloud", "Set up dbt", "Develop with dbt Cloud", "Build dbt projects", "About dbt projects", "dbt tips and tricks", "Build your DAG", "Models", "Tests", "Data tests", and "Unit tests". The main content area has a breadcrumb trail: `Home > Build dbt projects > Build your DAG > Tests`. The title "Add data tests to your DAG" is prominently displayed. Below it, a section "Related reference docs" lists links for "Test command", "Data test properties", "Data test configurations", and "Test selection examples". The "Overview" section explains that data tests are assertions on model results and provides an example of testing a column value. A right-hand sidebar titled "CONTENTS" lists sections like "Related reference docs", "Overview", "Singular data tests", "Generic data tests", "More generic data tests", "Example", "Storing test failures", and "FAQs". At the bottom of the right sidebar is a link to "Edit this page".

dbt

Docs Reference v1.7

Resources Community Search K Create a free account

Home > Build dbt projects > Build your DAG > Tests

## Add data tests to your DAG

### Related reference docs

- Test command
- Data test properties
- Data test configurations
- Test selection examples

### Overview

Data tests are assertions you make about your models and other resources in your dbt project (e.g. sources, seeds and snapshots). When you run `dbt test`, dbt will tell you if each test in your project passes or fails.

You can use data tests to improve the integrity of the SQL in each model by making assertions about the results generated. Out of the box, you can test whether a specified column in a model only contains non-null values, unique values, or values that have a corresponding value in another model (for example, a `customer_id` for an `order` corresponds to an `id` in the `customers` model), and values from a specified list. You can extend data tests to suit business logic specific to your organization – any assertion that you can make about your model in the form of a select query can be turned into a data test.

Data tests return a set of failing records. Generic data tests (f.k.a. schema tests) are defined using `test` blocks.

#### CONTENTS

- Related reference docs
- Overview
- Singular data tests
- Generic data tests
  - More generic data tests
- Example
- Storing test failures
- FAQs

Edit this page

# dbt test

Demo Session

# References

- <https://docs.getdbt.com/docs/introduction>





# Thank you!