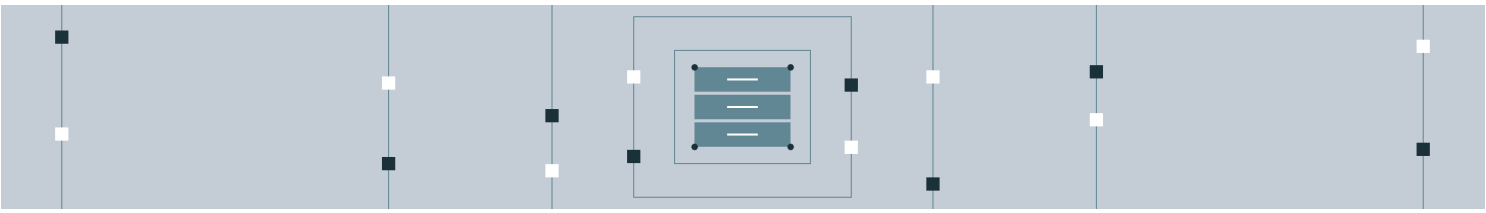


Loading a Data Warehouse Data Model in Real Time with the Databricks Lakehouse

Dimensional modeling implementation on the modern lakehouse using Delta Live Tables

by Leo Mao, Soham Bhatt and Abhishek Dey

November 7, 2022 in Platform Blog

Dimensional modeling is one of the most popular data modeling techniques for building a modern data warehouse. It allows customers to quickly develop facts and dimensions based on business needs for an enterprise. When helping customers in the field, we found many are looking for best practices and implementation reference architecture from Databricks.

In this article, we aim to dive deeper into the best practice of dimensional modeling on Databricks' Lakehouse Platform and provide a live example to load an EDW dimensional model in real-time using Delta Live Tables.

Here are the high-level steps we will cover in this blog:

1. Define a business problem
2. Design a dimensional model
3. Best practices and recommendations for dimensional modeling
4. Implementing a dimensional model in a Databricks Lakehouse
5. Conclusion

Dimensional modeling is business-oriented; it always starts with a business problem. Before building a dimensional model, we need to understand the business problem to solve, as it indicates how the data asset will be presented and consumed by end users. We need to design the data model to support more accessible and faster queries.

The Business Matrix is a fundamental concept in Dimensional Modeling, below is an example of the business matrix, where the columns are shared dimensions and rows represent business processes. The defined business problem determines the grain of the fact data and required dimensions. The key idea here is that we could incrementally build additional data assets with ease based on the Business Matrix and its shared or conformed dimensions.

	SHARED DIMENSIONS									
BUSINESS PROCESSES	Date	Customer	Product	Vendor	Promotion	Reseller	Sales Territory	Employee	Account	Organization
Internet Sales	✓	✓	✓	✓	✓		✓			
Reseller Sales	✓		✓		✓	✓	✓	✓		
General Ledger	✓								✓	✓
Sales Plan	✓		✓				✓			
Inventory	✓		✓	✓					✓	
Customer Surveys	✓	✓								
Customer Service Calls	✓	✓	✓					✓		

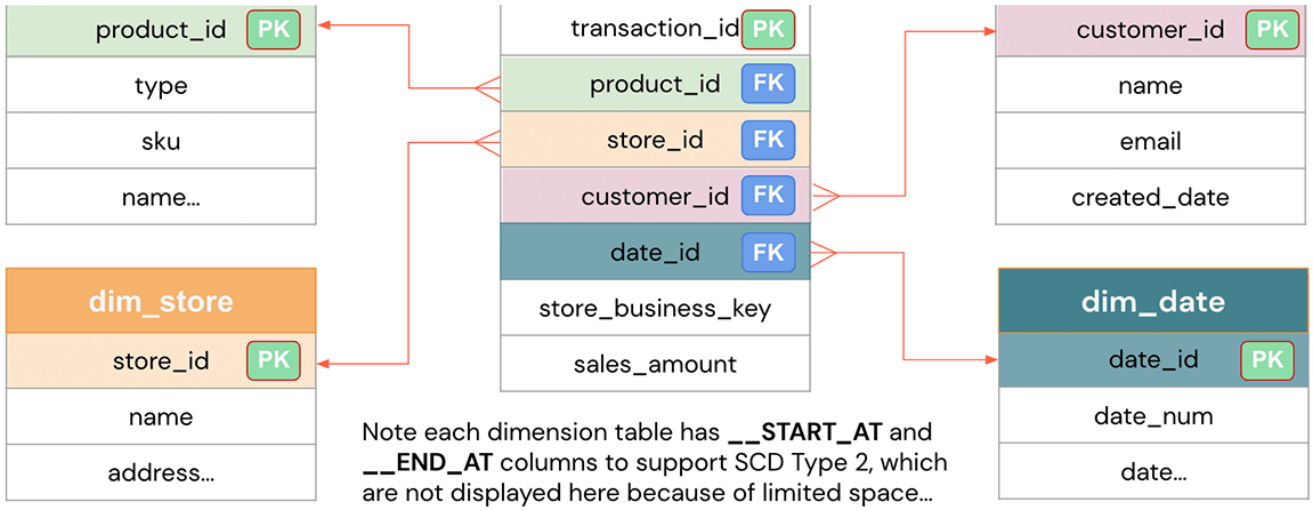
A Business Matrix with Shared Dimensions and Business Processes

Here we assume that the business sponsor would like to team to build a report to give insights on:

1. What are the top selling products so they can understand product popularity
2. What are the best performing stores to learn good store practices

2. Design a dimensional model

Based on the defined business problem, the data model design aims to represent the data efficiently for reusability, flexibility and scalability. Here is the high-level data model that could solve the business questions above.



Dimensional Model on the Lakehouse

The design should be easy to understand and efficient with different query patterns on the data. From the model, we designed the sales fact table to answer our business questions; as you can see, other than the foreign keys (FKs) to the dimensions, it only contains the numeric metrics used to measure the business, e.g. sales_amount.

We also designed dimension tables such as Product, Store, Customer, Date that provide contextual information on the fact data. Dimension tables are typically joined with fact tables to answer specific business questions, such as the most popular products for a given month, which stores are the best-performing ones for the quarter, etc.

3. Best practices and recommendations for dimensional modeling

With the Databricks Lakehouse Platform, one can easily design & implement dimensional models, and simply build the facts and dimensions for the given subject area.

Below are some of the best practices recommended while implementing a dimensional model:

- One should denormalize the dimension tables. Instead of the third normal form or snowflake type of model, dimension tables typically are highly denormalized with flattened many-to-one relationships within a single dimension table.
- Use conformed dimension tables when attributes in different dimension tables have the same column names and domain contents. This advantage is that data from different fact tables can be combined in a single report using conformed dimension attributes associated with each fact table.
- A usual trend in dimension tables is around tracking changes to dimensions over time to support as-is or as-was reporting. You can easily apply the following basic techniques for handling dimensions based on different requirements.

tracking over time.

This can be easily achieved out of the box with Delta Live Tables implementation.

- One can easily perform SCD type 1 or SCD type 2 using Delta Live Tables using APPLY CHANGES INTO

- **Primary + Foreign Key Constraints** allow end users like yourselves to understand relationships between tables.
- **Usage of IDENTITY Columns** automatically generates unique integer values when new rows are added. Identity columns are a form of surrogate keys. Refer to the blog link for more details.
- **Enforced CHECK Constraints** to never worry about data quality or data correctness issues sneaking up on you.

4. Implementing a dimensional model in a Databricks Lakehouse

Now, let us look at an example of Delta Live Tables based dimensional modeling implementation:

The example code below shows us how to create a dimension table (dim_store) using SCD Type 2, where change data is captured from the source system.

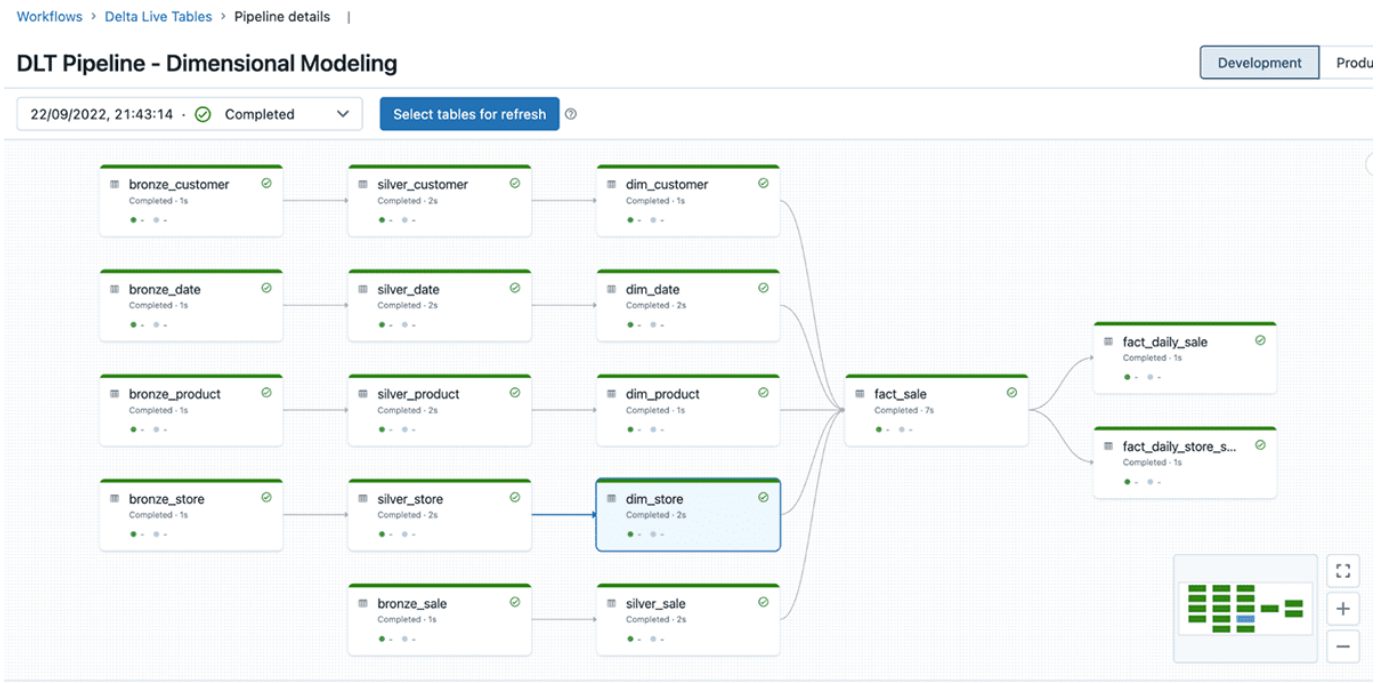
```
-- create the gold table
CREATE INCREMENTAL LIVE TABLE dim_store
TBLPROPERTIES ("quality" = "gold")
COMMENT "Slowly Changing Dimension Type 2 for store dimension in the gold layer";

-- store all changes as SCD2
APPLY CHANGES INTO live.dim_store
FROM STREAM(live.silver_store)
  KEYS (store_id)
  SEQUENCE BY updated_date
  COLUMNS * EXCEPT (_rescued_data, input_file_name)
  STORED AS SCD TYPE 2;
```

The example code below shows us how to create a fact table (fact_sale), with the constraint of **valid_product_id** we are able to ensure all fact records that are loaded have a valid product associated with it.

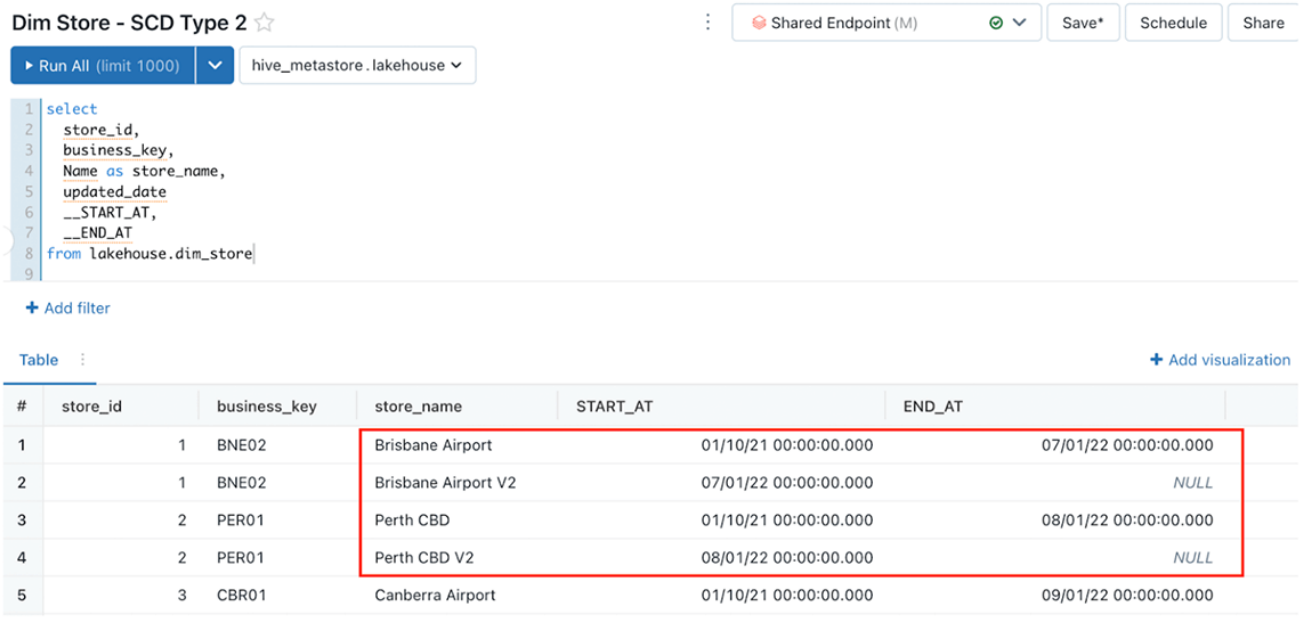
```
-- create the fact table for sales in gold layer
CREATE STREAMING LIVE TABLE fact_sale (
  CONSTRAINT valid_store_business_key EXPECT (store_business_key IS NOT NULL) ON VIOLATI
```

```
TBLPROPERTIES ("quality" = "gold", "ignoreChanges" = "true")
COMMENT "sales fact table in the gold layer" AS
  SELECT
    sale.transaction_id,
    date.date_id,
    customer.customer_id,
    product.product_id AS product_id,
    store.store_id,
    store.business_key AS store_business_key,
    sales_amount
  FROM STREAM(live.silver_sale) sale
  INNER JOIN live.dim_date date
  ON to_date(sale.transaction_date, 'M/d/yy') = to_date(date.date, 'M/d/yyyy')
  -- only join with the active customers
  INNER JOIN (SELECT * FROM live.dim_customer WHERE __END_AT IS NULL) customer
  ON sale.customer_id = customer.customer_id
  -- only join with the active products
  INNER JOIN (SELECT * FROM live.dim_product WHERE __END_AT IS NULL) product
  ON sale.product = product.SKU
  -- only join with the active stores
  INNER JOIN (SELECT * FROM live.dim_store WHERE __END_AT IS NULL) store
  ON sale.store = store.business_key
```

The Delta Live Table pipeline example could be found here. Please refer to Delta Live Tables quickstart on how to create a Delta Live Table pipeline. As seen below, DLT offers full visibility of the ETL pipeline and dependencies between different objects across bronze, silver, and gold layers following the lakehouse medallion architecture.



Here is an example of how the dimension table **dim_store** gets updated based on the incoming changes. Below, the Store **Brisbane Airport** was updated to **Brisbane Airport V2**, and with the out-of-box SCD Type 2 support, the original record ended on Jan 07 2022, and a new record was created which starts on the same day with an open end date (NULL) - which indicates the latest record for the Brisbane airport.



#	store_id	business_key	store_name	START_AT	END_AT
1	1	BNE02	Brisbane Airport	01/10/21 00:00:00.000	07/01/22 00:00:00.000
2	1	BNE02	Brisbane Airport V2	07/01/22 00:00:00.000	*NULL*
3	2	PER01	Perth CBD	01/10/21 00:00:00.000	08/01/22 00:00:00.000
4	2	PER01	Perth CBD V2	08/01/22 00:00:00.000	*NULL*
5	3	CBR01	Canberra Airport	01/10/21 00:00:00.000	09/01/22 00:00:00.000

SCD Type 2 for Store Dimension

For more implementation details, please refer to here for the full notebook example.

5. Conclusion

In this blog, we learned about dimensional modeling concepts in detail, best practices, and how to implement them using Delta Live Tables.

Learn more about dimensional modeling at Kimball Technology.

Get started on building your dimensional models in the Lakehouse

Try Databricks free for 14 days.

Related posts

Five Simple Steps for Implementing a Star Schema in Databricks With Delta Lake

May 20, 2022 by Cary Moore, Lucas Bilbro and Brenner Heintz in Product

Most data warehouse developers are very familiar with the ever-present star schema. Introduced by Ralph Kimball in the 1990s, a star schema is...

Data Modeling Best Practices & Implementation on a Modern Lakehouse

October 20, 2022 by Leo Mao, Abhishek Dey, Justin Breese and Soham Bhatt in Platform Blog

A Large number of our customers are migrating their legacy data warehouses to Databricks Lakehouse as it enables them to modernize not only...

Identity Columns to Generate Surrogate Keys Are Now Available in a Lakehouse Near You!

August 8, 2022 by Franco Patano in Product

What is an identity column? An identity column is a column in a database that automatically generates a unique ID number for each...

databricks

Why Databricks

Product

Solutions

Resources

About

Databricks Inc.
160 Spear Street, 13th Floor
San Francisco, CA 94105
1-866-330-0121

See Careers
at Databricks