7. Slowly Changing Dimensions (SCD)

Understanding SCD Types

Slowly Changing Dimensions (SCD) are dimensions that change slowly over time, rather than changing on a regular schedule or frequently. There are several types of SCDs:

1. **Type 0 (No Change)**: Historical changes are not tracked.
2. **Type 1 (Overwrite)**: Only the current values are maintained; historical values are overwritten.
3. **Type 2 (Add New Row)**: Historical changes are tracked by adding new rows with effective dates.
4. **Type 3 (Add New Attributes)**: Limited historical values are maintained in additional columns.
5. **Type 4 (History Table)**: Current values in main table, historical values in separate history table.
6. **Type 6 (Hybrid/Combined Approach)**: Combines Types 1, 2, and 3.

Implementing SCD in DBT

DBT provides multiple ways to implement SCDs, with Type 2 being the most common.

Type 1 SCD (Overwrite)

Type 1 is simple to implement with standard incremental models:

sql

```
{{ config(
    materialized = 'table',
    unique_key = 'customer_id'
) }}

SELECT
    customer_id,
```

```
    first_name,
    last_name,
    email,
    phone,
    address,
    updated_at
FROM {{ source('raw', 'customers') }}
```

Type 2 SCD (Historical Tracking)

For Type 2, DBT's snapshot feature is the recommended approach (covered in the next section). But you can also implement it manually:

sql

```
{{ config(
    materialized = 'incremental',
    unique_key = 'surrogate_key'
) }}

WITH source AS (
    SELECT
        customer_id,
        first_name,
        last_name,
        email,
        phone,
        address,
        updated_at
    FROM {{ source('raw', 'customers') }}
),

{% if is_incremental() %}
existing AS (
    SELECT
        surrogate_key,
        customer_id,
        first_name,
        last_name,
        email,
        phone,
        address,
        valid_from,
        valid_to
```

```
    FROM {{ this }}
    WHERE valid_to IS NULL
),

updates AS (
    SELECT
        source.customer_id,
        source.first_name,
        source.last_name,
        source.email,
        source.phone,
        source.address,
        source.updated_at,
        existing.surrogate_key,
        existing.valid_from
    FROM source
    JOIN existing ON source.customer_id = existing.customer_id
    WHERE (
        source.first_name != existing.first_name OR
        source.last_name != existing.last_name OR
        source.email != existing.email OR
        source.phone != existing.phone OR
        source.address != existing.address
    )
),

-- Close existing records that have changed
closing AS (
    SELECT
        surrogate_key,
        customer_id,
        first_name,
        last_name,
        email,
        phone,
        address,
        valid_from,
        updated_at AS valid_to
    FROM updates
    JOIN existing ON updates.surrogate_key = existing.surrogate_key
),

-- Create new records for the updated values
new_records AS (
    SELECT
        {{ dbt_utils.generate_surrogate_key(['customer_id', 'updated_at'])
```

```
}} AS surrogate_key,
        customer_id,
        first_name,
        last_name,
        email,
        phone,
        address,
        updated_at AS valid_from,
        NULL AS valid_to
    FROM updates
),

-- Find completely new customers
new_customers AS (
    SELECT
        source.customer_id,
        source.first_name,
        source.last_name,
        source.email,
        source.phone,
        source.address,
        source.updated_at
    FROM source
    LEFT JOIN existing ON source.customer_id = existing.customer_id
    WHERE existing.customer_id IS NULL
),

-- Generate surrogate keys for new customers
new_customer_records AS (
    SELECT
        {{ dbt_utils.generate_surrogate_key(['customer_id', 'updated_at'])
}} AS surrogate_key,
        customer_id,
        first_name,
        last_name,
        email,
        phone,
        address,
        updated_at AS valid_from,
        NULL AS valid_to
    FROM new_customers
),

-- Combine all records to insert
inserting AS (
    SELECT * FROM closing
```

```
    UNION ALL
    SELECT * FROM new_records
    UNION ALL
    SELECT * FROM new_customer_records
)

SELECT * FROM inserting

{% else %}

-- Initial load of all customers
initial_load AS (
    SELECT
        {{ dbt_utils.generate_surrogate_key(['customer_id', 'updated_at'])
}} AS surrogate_key,
        customer_id,
        first_name,
        last_name,
        email,
        phone,
        address,
        updated_at AS valid_from,
        NULL AS valid_to
    FROM source
)

SELECT * FROM initial_load

{% endif %}
```