

## Exercise - Explanation 1

- In this exercise, you will implement an SCD type 7 schema
- This exercise is comprehensive and you will have to use many of the things you have learned in the SQL in practice lectures
- The exercise can be broken up into three phases
- 1 Create the schema given on slide 19 (either in your own containerized PostgreSQL, or in the personal database I have made available to you) and populate it with the values given on the slide.
  - Remember to use appropriate data types when creating tables, and also specify the appropriate constraints, e.g., foreign keys. The SQL script 'create\_tables\_for\_update\_delete\_trigger' contains many useful commands.
  - Hint: the SQL function 'current\_date' will return the current date, and behaves similarly to the functions 'current\_timestamp(0)' or 'now()', when setting default column values, or when inserting values.

17 / 19

## Exercise

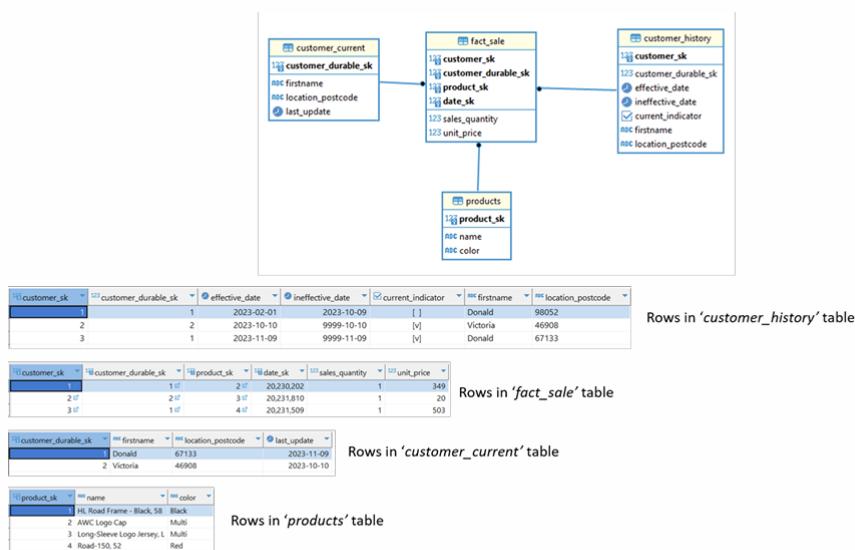


Figure: Schema bi\_trigger

19 / 19



Create table with  
junction.sql

## Exercise - Explanation 2

- 2 Create a trigger function and the trigger
  - Create a trigger function such that updates to the 'customer\_current' table trigger a new row to be written in 'customer\_history'.
  - Hint: You can use parts of the SQL script 'Update, Delete and Triggers' and modify it for this exercise.
- 3 Make updates to the 'location\_postcode' column in the 'customer\_current' table:
  - 3.1 Update 'Donalds' Postcode to 32584
  - 3.2 Update 'Victorias' Postcode to 73611
  - Check that the updates is implemented in the 'customer\_current' table, and that the 'customer\_history' table has been updated appropriately.

```

❶# create a trigger function and the trigger
# create a trigger function such that updates to the 'customer_current' table trigger a new row to be written in 'customer_history'

# Step 1.create the trigger

❷CREATE OR REPLACE FUNCTION bi_trigger.log_customer_changes()
RETURNS TRIGGER AS $$$
BEGIN
  -- Insert the old row into customer_history
  INSERT INTO bi_trigger.customer_history (
    customer_durable_sk,
    effective_date,
    ineffective_date,
    current_indicator,
    firstname,
    location_postcode
  )
  VALUES (
    OLD.customer_durable_sk,
    OLD.last_update,          -- effective_date from current row   The old row became effective on the date it was last updated.
    CURRENT_DATE,             -- set ineffective date to today   Marks today as the date when the old row stopped being current.
    NULL,                     -- mark old as non-current   Marks the old row as not current.
    OLD.firstname,            -- Copy the old values.
    OLD.location_postcode    -- Copy the old values.
  );
  -- Update the last_update of new row to current date
  NEW.last_update := CURRENT_DATE;           NEW is the row that is being updated in customer_current.
  This line modifies the value of the last_update column in that row before the update is applied.
  It does not insert a new row, it simply sets the column value of the existing row to the current date.
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

❸# Step 2: Create the trigger
CREATE TRIGGER trg_customer_update _____ Just the name of the trigger
BEFORE UPDATE ON bi_trigger.customer_current _____ The trigger will be executed when an update in customer_current is happening
FOR EACH ROW
EXECUTE FUNCTION bi_trigger.log_customer_changes();

```

customer_history 1							
	customer_sk	customer_durable_sk	effective_date	ineffective_date	current_indicator	firstname	location_postcode
Grid	1	1	2023-02-01	2023-10-09 [NULL]		Donald	98052
Text	2	2	2023-10-10	9999-10-10 [v]		Victoria	46908
Text	3	3	2023-11-09	9999-11-09 [v]		Donald	67133

Nach dem update

customer_history 1							
	customer_sk	customer_durable_sk	effective_date	ineffective_date	current_indicator	firstname	location_postcode
Grid	1	1	2023-02-01	2023-10-09 [NULL]		Donald	98052
Text	2	2	2023-10-10	9999-10-10 [v]		Victoria	46908
Text	3	3	2023-11-09	9999-11-09 [v]		Donald	67133
Text	4	1	2023-11-09	2025-10-25 [NULL]		Donald	67133
Text	5	2	2023-10-10	2025-10-25 [NULL]		Victoria	46908



Trigger und Update.sql