Course project

“Garden Cantre”

Katsiaryna Tsikhanovich

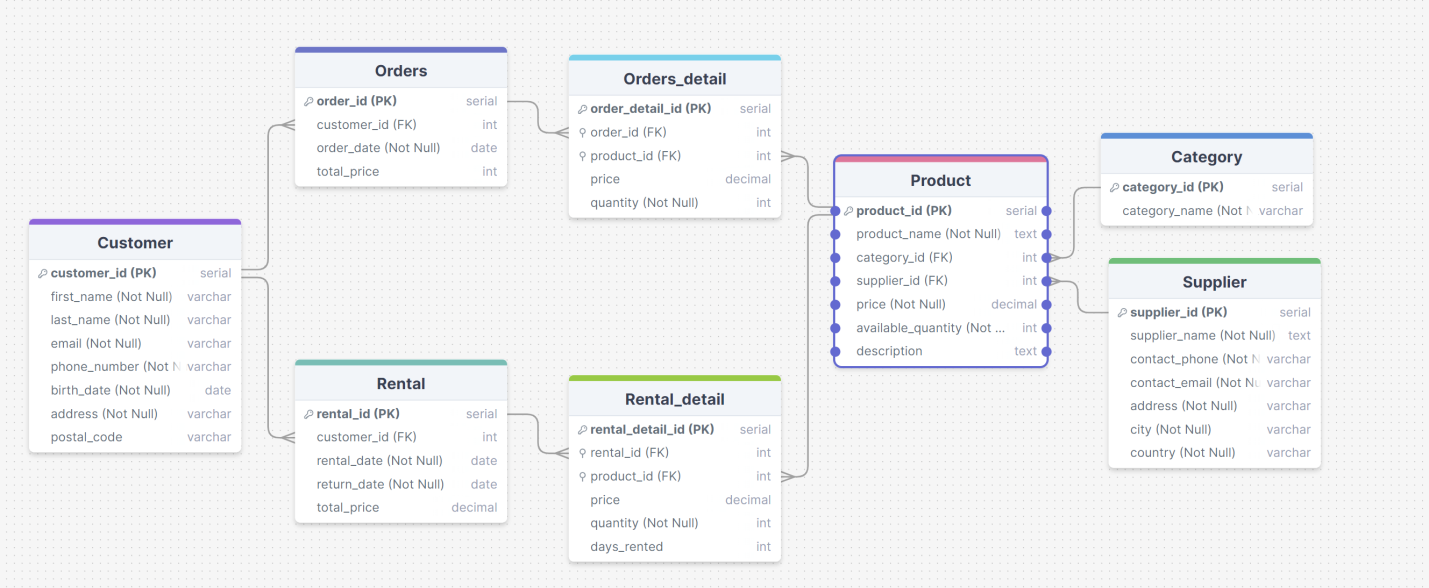
2024

1. *BUSINESS DESCRIPTION*

Garden Centre is a little shop in London where you can buy indoor plants, trees, seeds, gardening tools, soil and fertilizers. A gardening equipment rental service is also available. We have a lot of suppliers from different countries. Our mission is to make gardening your joy and happiness.

1. *MODEL DESCRIPTIONS*

*ER-SCHEMA FOR OLTP:*

**

Script: Creation\_and\_functions\_OLTP\_Tsikhanovich.sql

#### Tables and Relationships

1. **Customer**
   * **Fields**:
     + customer\_id (Primary Key): A unique identifier for each customer.
     + first\_name (Not Null): The first name of the customer.
     + last\_name (Not Null): The last name of the customer.
     + email (Not Null): The email address of the customer.
     + phone\_number (Not Null): The phone number of the customer.
     + birth\_date (Not Null): The birth date of the customer.
     + address (Not Null): The physical address of the customer.
     + postal\_code: The postal code of the customer’s address.
   * **Relationships**:
     + The Customer table is referenced by the Orders and Rental tables via the customer\_id foreign key, establishing a one-to-many relationship with both tables.
2. **Supplier**
   * **Fields**:
     + supplier\_id (Primary Key): A unique identifier for each supplier.
     + supplier\_name (Not Null): The name of the supplier.
     + contact\_phone (Not Null, Unique): The contact phone number of the supplier.
     + contact\_email (Not Null, Unique): The contact email of the supplier.
     + address (Not Null): The address of the supplier.
     + city (Not Null): The city where the supplier is located.
     + country (Not Null): The country where the supplier is located.
   * **Relationships**:
     + The Supplier table is referenced by the Product table via the supplier\_id foreign key, establishing a one-to-many relationship with the Product table.
3. **Category**
   * **Fields**:
     + category\_id (Primary Key): A unique identifier for each product category.
     + category\_name (Not Null, Unique): The name of the product category.
   * **Relationships**:
     + The Category table is referenced by the Product table via the category\_id foreign key, establishing a one-to-many relationship with the Product table.
4. **Product**
   * **Fields**:
     + product\_id (Primary Key): A unique identifier for each product.
     + product\_name (Not Null, Unique): The name of the product.
     + category\_id (Foreign Key): References the category\_id in the Category table.
     + supplier\_id (Foreign Key): References the supplier\_id in the Supplier table.
     + price (Not Null): The price of the product.
     + available\_quantity (Not Null): The available quantity of the product in stock.
     + description: A description of the product.
   * **Relationships**:
     + The Product table has a many-to-one relationship with both the Supplier and Category tables.
     + It is also referenced by the Orders\_detail and Rental\_detail tables via the product\_id foreign key, establishing a one-to-many relationship with these tables.
5. **Orders**
   * **Fields**:
     + order\_id (Primary Key): A unique identifier for each order.
     + customer\_id (Foreign Key): References the customer\_id in the Customer table.
     + order\_date (Not Null): The date the order was placed.
     + total\_price: The total price of the order.
   * **Relationships**:
     + The Orders table has a many-to-one relationship with the Customer table.
     + It is referenced by the Orders\_detail table via the order\_id foreign key, establishing a one-to-many relationship with the Orders\_detail table.
6. **Orders\_detail**
   * **Fields**:
     + order\_detail\_id (Primary Key): A unique identifier for each order detail.
     + order\_id (Foreign Key): References the order\_id in the Orders table.
     + product\_id (Foreign Key): References the product\_id in the Product table.
     + price (Decimal): The price of the product in the order.
     + quantity (Not Null): The quantity of the product in the order.
   * **Relationships**:
     + The Orders\_detail table has a many-to-one relationship with both the Orders and Product tables.
7. **Rental**
   * **Fields**:
     + rental\_id (Primary Key): A unique identifier for each rental.
     + customer\_id (Foreign Key): References the customer\_id in the Customer table.
     + rental\_date (Not Null): The date the rental started.
     + return\_date (Not Null): The date the rental ended.
     + total\_price (Decimal): The total price of the rental.
   * **Relationships**:
     + The Rental table has a many-to-one relationship with the Customer table.
     + It is referenced by the Rental\_detail table via the rental\_id foreign key, establishing a one-to-many relationship with the Rental\_detail table.
8. **Rental\_detail**
   * **Fields**:
     + rental\_detail\_id (Primary Key): A unique identifier for each rental detail.
     + rental\_id (Foreign Key): References the rental\_id in the Rental table.
     + product\_id (Foreign Key): References the product\_id in the Product table.
     + price (Decimal): The price of the product in the rental.
     + quantity (Not Null): The quantity of the product in the rental.
     + days\_rented: The number of days the product was rented.
   * **Relationships**:
     + The Rental\_detail table has a many-to-one relationship with both the Rental and Product tables.

### Summary

* **One-to-Many Relationships**:
  + Customer to Orders
  + Customer to Rental
  + Supplier to Product
  + Category to Product
  + Orders to Orders\_detail
  + Product to Orders\_detail
  + Rental to Rental\_detail
  + Product to Rental\_detail

This schema allows the business to manage and track products, orders, rentals, suppliers, and customers effectively. It supports detailed transactional data collection and can be extended to support various reporting and analytical needs.

#### Functions: 1) validate\_rental\_dates

**Purpose**: The function validate\_rental\_dates is designed to ensure that the rental start date (rental\_date) is earlier than the rental end date (return\_date). This validation is crucial for maintaining data integrity and ensuring that the rental periods are logically consistent.

#### 2) calculate\_rental\_days

**Purpose**: The function calculate\_rental\_days calculates the number of days between the rental start date (rental\_date) and the rental end date (return\_date). This function is useful for determining the duration of a rental period.

*3.**PREPARE SCRIPT TO LOAD DATA FROM CSV TO THE OLTP DATABASE*

Script: Insertion\_csv\_files\_Tsikhanovich.sql

### Overall Description of the Function load\_data\_from\_csv

#### Purpose

The load\_data\_from\_csv function is designed to automate the process of loading data from multiple CSV files into respective tables in a PostgreSQL database. It ensures that the data is properly loaded, validated, and cleaned before being used in further operations. This function simplifies the data ingestion process by handling multiple steps including data loading, validation, and calculation.

#### Parameters

The function accepts the file paths of eight different CSV files as input parameters:

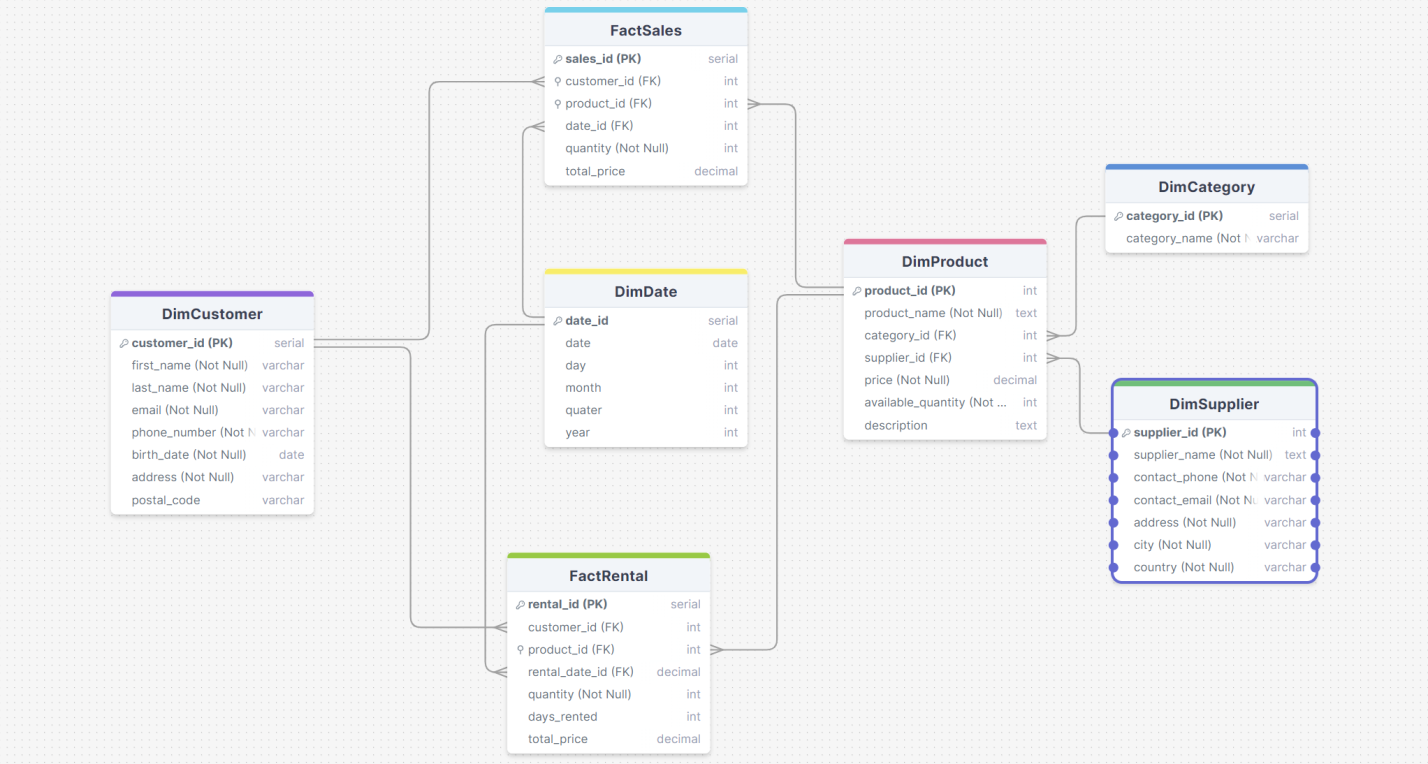
* customer\_file\_path (TEXT): Path to the CSV file containing customer data.
* order\_file\_path (TEXT): Path to the CSV file containing order data.
* order\_detail\_file\_path (TEXT): Path to the CSV file containing order details data.
* supplier\_file\_path (TEXT): Path to the CSV file containing supplier data.
* rental\_file\_path (TEXT): Path to the CSV file containing rental data.
* rental\_detail\_file\_path (TEXT): Path to the CSV file containing rental details data.
* product\_file\_path (TEXT): Path to the CSV file containing product data.
* category\_file\_path (TEXT): Path to the CSV file containing category data.

#### Function Logic

1. **Load Data into Main Tables**:
   * The function uses the COPY command to load data from each CSV file into the corresponding table in the database.
   * The COPY command is dynamically constructed using the format function and executed using the EXECUTE statement.
   * The data is expected to be in CSV format with a comma delimiter and a header row. The encoding used is LATIN1.
2. **Data Validation and Cleanup**:
   * **Email Validation**: Invalid email addresses are removed from the Customer and Supplier tables using regular expressions to match valid email formats.
   * **Product Validation**: Products with non-positive prices or negative available quantities are removed from the Product table.
   * **Quantity Validation**: Entries in the Rental\_detail and Orders\_detail tables with non-positive quantities are removed.
   * **Date Validation**: Entries in the Rental and Orders tables with dates earlier than January 1, 2011, are removed. Customers with birth dates earlier than January 1, 1945, are removed.
   * **Rental Date Validation**: Rentals where the rental\_date is not earlier than the return\_date are removed using the validate\_rental\_dates function.
3. **Insert Calculated Data**:
   * **Calculate Rental Days**: The days\_rented field in the Rental\_detail table is updated based on the difference between rental\_date and return\_date using the calculate\_rental\_days function.
   * **Calculate Prices**:
     + The price field in the Orders\_detail table is calculated as the product of the unit price and the quantity ordered.
     + The price field in the Rental\_detail table is calculated as the product of the unit price, the quantity rented, and the number of days rented.
   * **Calculate Total Order Price**: The total\_price field in the Orders table is updated based on the sum of prices for all items in the order.
4. **Execution**:
   * The function is executed using a SELECT statement where appropriate file paths are provided as arguments.
   * After executing the function, queries can be run to view the data in each table.

*4. Design schemas Create needed tables, indexes*

*ER-Schema for OLAP:*

**

*Script: Creation\_OLAP\_Tsikhanovich.sql*

1. *PREPARE SCRIPT TO LOAD DATA FROM THE OLTP DATABASE TO THE OLAP DATABASE*

Sciprt: Load\_data\_to\_OLAP\_Tsikhanovich.sql

This script is designed to facilitate data transfer from an OLTP (Online Transaction Processing) database to an OLAP (Online Analytical Processing) database using PostgreSQL Foreign Data Wrapper (FDW). It sets up the necessary connections, imports the schema, and defines a function to transfer and transform the data into the appropriate dimensional and fact tables. Below is a detailed description of each step in the script.

#### 1. Install Required Extension

CREATE EXTENSION IF NOT EXISTS postgres\_fdw;

* **Purpose**: Install the PostgreSQL Foreign Data Wrapper (FDW) extension if it is not already installed. This extension allows PostgreSQL to access data stored in external PostgreSQL databases.

#### 2. Create Foreign Server

CREATE SERVER oltp\_server

FOREIGN DATA WRAPPER postgres\_fdw

OPTIONS (host 'localhost', dbname '01', port '5432');

* **Purpose**: Define a foreign server named oltp\_server that connects to the OLTP database located on localhost with the database name 01 and port 5432.

#### 3. Create User Mapping

CREATE USER MAPPING FOR CURRENT\_USER

SERVER oltp\_server

OPTIONS (user 'postgres', password '159357');

* **Purpose**: Create a user mapping for the current PostgreSQL user to connect to the oltp\_server with the specified username (postgres) and password (159357).

#### 4. Import Schema from Foreign Server

IMPORT FOREIGN SCHEMA public

FROM SERVER oltp\_server

INTO public;

* **Purpose**: Import the schema from the public schema of the oltp\_server into the public schema of the local database. This allows the local database to access tables from the remote OLTP database.

#### 5. Create Data Transfer Function

DROP FUNCTION IF EXISTS transferring\_data CASCADE;

CREATE OR REPLACE FUNCTION transferring\_data()

RETURNS void AS $$

BEGIN

-- Data transfer logic here

END;

$$ LANGUAGE plpgsql;

* **Purpose**: Define a function named transferring\_data to transfer data from the OLTP database to the OLAP database. The function performs the following operations:

#### 6. Data Transfer Operations

**DimCustomer**:

INSERT INTO DimCustomer (first\_name, last\_name, email, phone\_number, birth\_date, address, postal\_code)

SELECT c.first\_name, c.last\_name, c.email, c.phone\_number, c.birth\_date, c.address, c.postal\_code

FROM public.customer c

LEFT JOIN DimCustomer dc ON c.email = dc.email

WHERE dc.customer\_id IS NULL;

* **Purpose**: Transfer new customer data from the OLTP customer table to the OLAP DimCustomer table, avoiding duplicates based on the email address.

**DimSupplier**:

INSERT INTO DimSupplier (supplier\_name, contact\_phone, contact\_email, address, city, country)

SELECT s.supplier\_name, s.contact\_phone, s.contact\_email, s.address, s.city, s.country

FROM public.supplier s

LEFT JOIN DimSupplier ds ON s.contact\_email = ds.contact\_email

WHERE ds.supplier\_id IS NULL;

* **Purpose**: Transfer new supplier data from the OLTP supplier table to the OLAP DimSupplier table, avoiding duplicates based on the contact email.

**DimCategory**:

INSERT INTO DimCategory (category\_name)

SELECT c.category\_name

FROM public.category c

LEFT JOIN DimCategory dc ON c.category\_name = dc.category\_name

WHERE dc.category\_id IS NULL;

* **Purpose**: Transfer new category data from the OLTP category table to the OLAP DimCategory table, avoiding duplicates based on the category name.

**DimProduct**:

INSERT INTO DimProduct (product\_name, category\_id, supplier\_id, price, available\_quantity, description, start\_date, end\_date, current\_flag)

SELECT p.product\_name, dc.category\_id, ds.supplier\_id, p.price, p.available\_quantity, p.description, NOW(), '9999-12-31', TRUE

FROM public.product p

JOIN public.category c ON p.category\_id = c.category\_id

JOIN DimCategory dc ON c.category\_name = dc.category\_name

JOIN public.supplier s ON p.supplier\_id = s.supplier\_id

JOIN DimSupplier ds ON s.contact\_email = ds.contact\_email

LEFT JOIN DimProduct dp ON p.product\_name = dp.product\_name AND dc.category\_id = dp.category\_id AND ds.supplier\_id = dp.supplier\_id

WHERE dp.product\_id IS NULL;

* **Purpose**: Transfer new product data from the OLTP product table to the OLAP DimProduct table, avoiding duplicates based on a combination of product name, category, and supplier.

**DimDate**:

INSERT INTO DimDate (date, day, month, quarter, year)

SELECT DISTINCT o.order\_date, EXTRACT(DAY FROM o.order\_date), EXTRACT(MONTH FROM o.order\_date), EXTRACT(QUARTER FROM o.order\_date), EXTRACT(YEAR FROM o.order\_date)

FROM public.orders o

LEFT JOIN DimDate dd ON o.order\_date = dd.date

WHERE dd.date IS NULL;

INSERT INTO DimDate (date, day, month, quarter, year)

SELECT DISTINCT r.rental\_date, EXTRACT(DAY FROM r.rental\_date), EXTRACT(MONTH FROM r.rental\_date), EXTRACT(QUARTER FROM r.rental\_date), EXTRACT(YEAR FROM r.rental\_date)

FROM public.rental r

LEFT JOIN DimDate dd ON r.rental\_date = dd.date

WHERE dd.date IS NULL;

* **Purpose**: Transfer unique order and rental dates from the OLTP orders and rental tables to the OLAP DimDate table, avoiding duplicates based on the date.

**FactSales**:

INSERT INTO FactSales (customer\_id, product\_id, date\_id, quantity, total\_price)

SELECT c.customer\_id, p.product\_id, d.date\_id, od.quantity, (od.quantity \* p.price) AS total\_price

FROM public.orders o

JOIN public.orders\_detail od ON o.order\_id = od.order\_id

JOIN public.customer c ON o.customer\_id = c.customer\_id

JOIN public.product p ON od.product\_id = p.product\_id

JOIN DimCustomer dc ON c.email = dc.email

JOIN DimProduct dp ON p.product\_name = dp.product\_name

JOIN DimDate d ON o.order\_date = d.date

LEFT JOIN FactSales fs ON c.customer\_id = fs.customer\_id AND p.product\_id = fs.product\_id AND d.date\_id = fs.date\_id

WHERE fs.sales\_id IS NULL;

* **Purpose**: Transfer new sales transaction data from the OLTP orders and orders\_detail tables to the OLAP FactSales table, avoiding duplicates based on customer, product, and date.

**FactRental**:

INSERT INTO FactRental (customer\_id, product\_id, rental\_date\_id, quantity, days\_rented, total\_price)

SELECT c.customer\_id, p.product\_id, d.date\_id, rd.quantity, rd.days\_rented, (rd.quantity \* p.price \* rd.days\_rented) AS total\_price

FROM public.rental r

JOIN public.rental\_detail rd ON r.rental\_id = rd.rental\_id

JOIN public.customer c ON r.customer\_id = c.customer\_id

JOIN public.product p ON rd.product\_id = p.product\_id

JOIN DimCustomer dc ON c.email = dc.email

JOIN DimProduct dp ON p.product\_name = dp.product\_name

JOIN DimDate d ON r.rental\_date = d.date

LEFT JOIN FactRental fr ON c.customer\_id = fr.customer\_id AND p.product\_id = fr.product\_id AND d.date\_id = fr.rental\_date\_id

WHERE fr.rental\_id IS NULL;

* **Purpose**: Transfer new rental transaction data from the OLTP rental and rental\_detail tables to the OLAP FactRental table, avoiding duplicates based on customer, product, and date.

#### 7. Execute the Data Transfer Function

SELECT transferring\_data();

* **Purpose**: Execute the transferring\_data function to perform the data transfer and transformation operations.

#### 8. Queries to View Data

SELECT \* FROM DimCustomer;

SELECT \* FROM DimSupplier;

SELECT \* FROM DimCategory;

SELECT \* FROM DimProduct;

SELECT \* FROM DimDate;

SELECT \* FROM FactSales;

SELECT \* FROM FactRental;

* **Purpose**: View the data in the dimensional and fact tables to verify that the data has been transferred and transformed correctly.

### Summary

This script sets up a connection to an external OLTP database using PostgreSQL FDW, imports the schema, and defines a function to transfer and transform data into the OLAP database. The function transferring\_data ensures that new records are inserted into the dimensional and fact tables, maintaining data integrity and avoiding duplicates. By running the provided queries, users can verify the data transfer and transformation process.

1. *HOW TO RUN THE PROJECT*

**Link to repository:** https://github.com/KatTihanovich/db\_course\_work

1. In PostgreSQL create two databases: Garden\_Centre and DWH.
2. Clone git repository with all necessary scripts to initialize databases, loading data into them and datasets;
3. In the database Garden\_Centre run Creation\_and\_functions\_OLTP\_Tsikhanovich.sql to initialize necessary tables and functions;
4. Run Insertion\_csv\_files\_Tsikhanovich.sql and put appropriate path to the csv files;   
   After that you should get 8 tables according to ER-SCHEMA FOR OLTP.
5. In the created DWH database run Creation\_OLAP\_Tsikhanovich.sql;  
    You should get tables according to ER-Schema for OLAP.
6. Run SDC2\_changes\_Tsikhanovich.sql;
7. Then make sure that the credentials for connection at the top of Load\_data\_to\_OLAP\_Tsikhanovich.sql are correct.

CREATE SERVER oltp\_server

FOREIGN DATA WRAPPER postgres\_fdw

OPTIONS (host 'localhost', dbname 'DWH', port '5432');

CREATE USER MAPPING FOR CURRENT\_USER

SERVER oltp

OPTIONS (user 'postgres', password 'put your password');

1. Run Load\_data\_to\_OLAP\_Tsikhanovich.sql;
2. Open PowerBi project