Diagram

Description automatically generated

# Creating a star schema

We begin with a **normalized database schema**, where information is stored across many related tables to avoid duplication. In our case, we had a structure where: Sales data is recorded in the SALES table. Information about customers, employees, stores, products, and locations is stored across several other tables like CUSTOMERS, EMPLOYEES, PRODUCTS, CATEGORIES, ADDRESSES, CITIES, COUNTRIES, etc.

To convert this schema to a star schema I followed these steps:

**Step 1: Picked the Fact Table**

The SALES table is chosen as the main fact table because it records key events — each sale, its amount, and when/where it happened.

**Step 2: Chose Dimension Tables**

I selected the following tables to describe the sales: CUSTOMERS, PRODUCTS, EMPLOYEES, STORES, DATE .

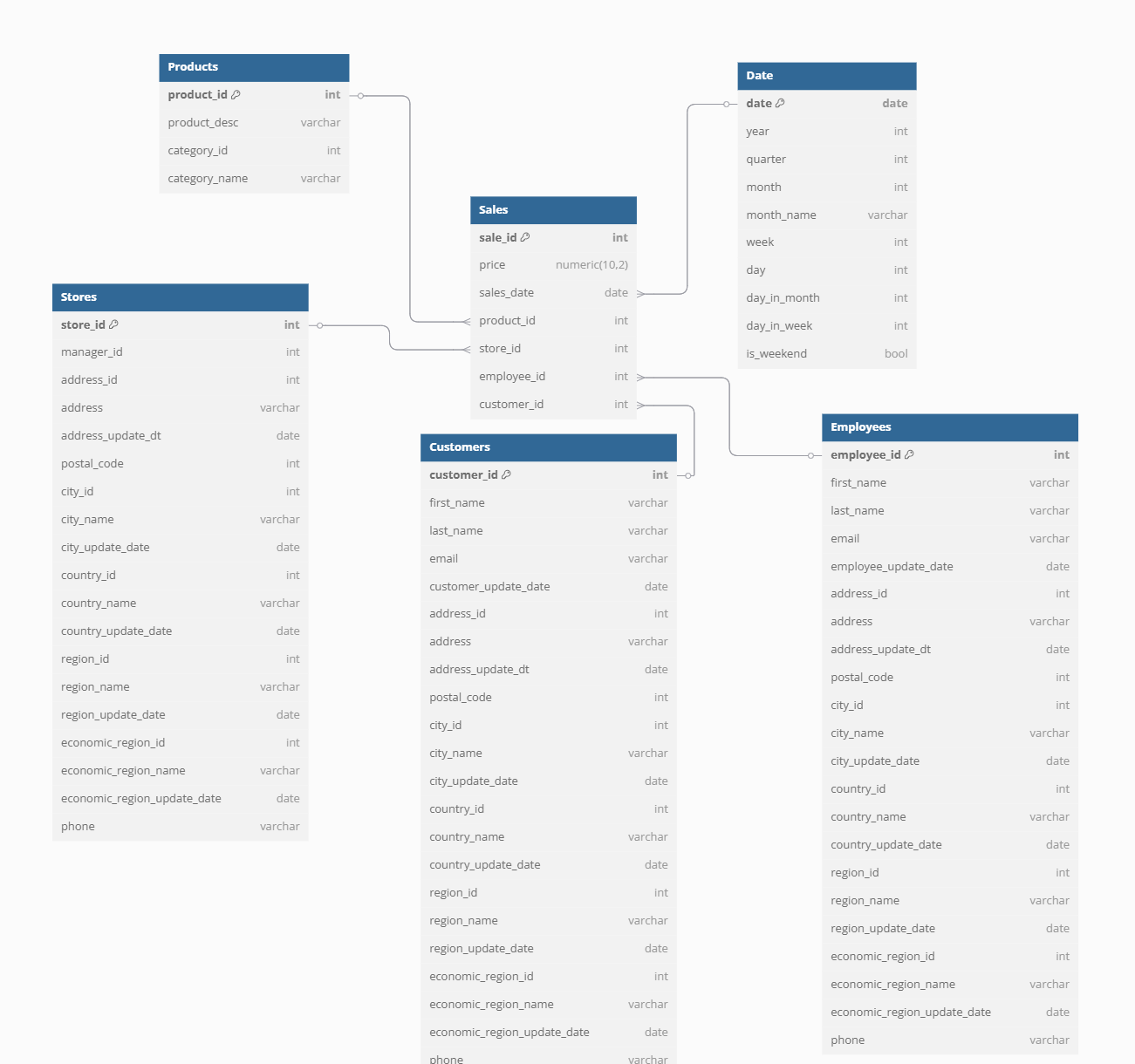
**Step 3: Combined Related Information**

To make reporting easier, I **combined** related tables into single, flat tables. For example:

1. In the CUSTOMERS table, we added details from ADDRESSES, CITIES, COUNTRIES, and REGIONS.
2. In the PRODUCTS table, we added the product category name directly.

**Step 4: Connected All to the Fact Table**

We linked all the dimension tables directly to the SALES table. The result looks like a star, with SALES in the center and dimensions around it — this is why it's called a **star schema.**

Result:

Transforming the star schema to a snowflake:

The star schema is simple and good for performance, but it can lead to data repetition. To avoid this, we can convert it into a **snowflake schema** by breaking large dimension tables into smaller related ones again. For the result I did the following:

**Step 1: Looked for a Repeated Information**

I saw that fields like city name, country name, region, etc., were repeated in several dimension tables. That means we can separate them out to avoid duplication.

**Step 2: Split Tables into Smaller Parts**

I took the large dimension tables and broke them into parts:

1. CUSTOMERS, EMPLOYEES, and STORES all use the ADDRESSES table.
2. ADDRESSES connects to CITIES, which connects to COUNTRIES, and so on.
3. We also separated CATEGORIES from PRODUCTS.

**Step 3: I used Foreign Keys Instead of Repeating Data**

Instead of keeping all location details in each table, I used foreign keys:

1. CUSTOMERS has address\_id instead of full address info.
2. PRODUCTS has category\_id instead of category name.

**Step 4: Built the Full Structure**

I connected all the smaller tables properly. This created a structure where dimensions are no longer flat, but split into related tables — this looks like a snowflake, so it’s called a **snowflake schema.**

Result:

