LinkedIn-https://www.linkedin.com/in/adeniran-olanrewaju/

A COMPREHENSIVE GUIDE TO DATABASE DESIGN: Case Study - Crocs (The Company)

TABLE OF CONTENTS

Introduction

This article will provide a general overview of how a Database can be effectively designed from start to finish to meet ever-evolving business needs. In the data landscape, one of the most sought-after technical skills is the ability to efficiently develop a scalable database design for your clients and customers while being a consultant. In this article, we want to build a new database for Crocs (company), hereby referred to as the client. Because a good database design is essential for achieving your goals when working with a database, Investing the time required to learn the principles of good design makes sense. In the end, both you as the consultant and the client are much more likely to end up with a database that meets your needs and can easily accommodate change.

What is considered a good Database Design?

A couple of universally acceptable principles exist that guide the database design process. One common guideline to prioritize is maintaining Data Integrity and accuracy, which in most cases helps to eliminate or avoid data redundancy in the database. Duplicate information and null entries within your records will increase the likelihood of errors, waste of storage, and inconsistencies. Another policy is ensuring business rule alignment by consistently providing valid, complete, and accurate information. If the database contains incorrect information, any reports based on it will also be inaccurate, leading to misinformed decisions.

Therefore, a good database design is one that:

- ➔ Maintain data integrity at all levels (field, table, and relationship levels) to ensure accuracy and completeness of information.
- ➔ Consistently aligns with relevant business rules, providing valid and meaningful information.
- ➔ Divide your information into subject-based tables to reduce redundant data.

➔ Scalability - design the database structure to facilitate future modifications and expansions in response to evolving business information requirements.

Common Terminologies in Database Design

We have many commonly used terms in the database design process. Getting familiar with some of these terms is a good practice, as they make the design process clearer and much easier to understand.

Let's take a look at these commonly used terms:

A. Table: A table is the chief structure of a database, representing specific subjects, objects, or events. They help organize data in the database.
B. Field: A field in a database is a single piece of data or a specific attribute within a table. It stores actual data.
C. Record: A record (row) in a database is a complete set of related fields representing one item or entity.
D. Keys: Keys in a database are attributes or combinations of attributes used to identify and establish relationships between tables uniquely.
E. Data: The value you store in a table. Data itself is meaningless.
F. Information: This is the organized and processed data stored in tables. Information is meaningful.
G. Primary Key: A primary key in a database is a unique identifier for each record in a table.
H. Foreign Key: A foreign key in a database is a field that links to the primary key of another table to establish a relationship between the two tables.
I. Relationships: Relationships in a database connect tables through keys to organize and manage related data.
J. Nulls: Nulls in a database represent missing or undefined values in a field.
K. Views: Views in a database are virtual tables created by querying and combining data from other tables.
L. Data Integrity: Data integrity in a database refers to the accuracy and consistency of stored data.

M. ERD (Entity Relationship Diagram): An ERD (Entity Relationship Diagram) in a database is a visual representation of the relationships between entities (tables) in a database.
N. Structured Query Language (SQL): Structured Query Language (SQL) is a programming language used to manage and manipulate relational databases.

Determining the Objective of the database (Interviews)

During the interview stage, we engage with users and management to define the purpose and objectives of our database project. Through conversations and inquiries, we gather insights to understand the specific needs, requirements, and expectations of the database. By this time, Key stakeholders are identified, and discussions focus on business processes, data dependencies, and desired outcomes. These interviews lay the groundwork for designing a database solution that aligns with organizational goals.

Interview Questions and Answers:

Question 1
Interviewer (Our Question): What specific challenges or inefficiencies are you currently facing with your data management processes?

Management (Answer): At Crocs, our data is scattered, manual processes slow us down, teams struggle to access what they need, errors creep in, and security worries loom. We need a smart solution to bring everything together, automate tasks, ensure accuracy, and keep our data safe.

Question 2
Interviewer (Our Question): What are the main objectives or outcomes you hope to achieve with this new database?

Management (Answer): At Crocs, our new database aims to bring all our data together, automate tasks, provide easy access, ensure accuracy, and enhance security, making life easier for everyone.

Question 3:
Interviewer (Our Question): How do you envision the database improving your overall business operations and decision-making processes?

Management (Answer): At Crocs, the new database will turbocharge our operations, provide instant access to crucial information, uncover insights, and serve as our guardian angel, guiding smarter decisions and propelling us forward.

Question 4
Interviewer (Our Question): Who are the key stakeholders or users of the database, and what are their specific needs or requirements?

Management (Answer): At Crocs, the database serves everyone—sales, product, finance teams, and customers. It's the go-to hub for quick access to information, accurate data, and seamless experiences, ensuring everyone's needs are met with a smile.

Final Mission Statement:

“At Crocs, our database is central to delivering exceptional customer experiences and driving business growth. We aim to maintain a secure, reliable, and scalable database that supports efficient operations, protects customer data, and provides actionable insights to innovate our product offerings and enhance our global supply chain”

Data collection analysis and organizing the necessary information

Now that the interviews are finished, we have gathered the needed information from the meetings and collated Data. This helps us create our initial characteristic list. The characteristics list will be reviewed later and used in compiling a complete list of fields (columns) for our table.

The characteristics list gathered:

I. Customer name, address, city, state, postal code, phone number, email address, country, and purchase date.
II. Product name, unit price, unit cost, category, stock quantity, ratings, SKU, description, and category names.
III. Order date, amount, ship date, discount, price, Quantity, and Order status.
IV. Store name, location, address, city, state, stock quantity, country, and phone number.
V. Employee name, phone, address, city, state, country, zip code, and department.

We now have our characteristics list, explored with the help of the data collection exercise and the interviews we embarked on earlier.

We are reviewing and compiling a complete list of fields from the characteristics list.

We will refine the characteristic lists above based on our analysis and interviews. We will eliminate duplicates, refine names, and separate each piece of information where required.

Below is our updated, complete list of fields (each with complete and accurate information):

We will name it our Preliminary Field List.

I. FirstName, LastName, Address, City, Region, PostalCode, Phone, EmailAddress, Country.
II. ProductName, UnitPrice, UnitCost, Category, StockQuantity, Ratings, SKU, Description, Category.
III. OrderDate, ShipDate, ShippingAddress, BillingAddress, TotalAmount, OrderStatus.
IV. Quantity, UnitPrice, Discount.
V. StoreName, Location, Address, City, State, StockQuantity, Country, Phone.
VI. EmployeeName, Phone, Address, City, Region, Country, PostalCode, Department.
VII. DepartmentName, Description.

Don’t include calculated data

In most cases, you should keep the results of calculations from tables (calculated fields). Instead, you can have this activity performed within the database when you want to see the results. For example: we can create a new calculated field list to add the TotalAmount field in our third preliminary field list above.

Completing our List of Fields — Reviewing Both Lists with Users and Management

Once we had determined our initial columns and calculated fields, we conducted a brief interview with users and management to review both the Preliminary Field List and the Calculated Field List. The goal is to ensure completeness and identify any omitted fields.

Dividing the information into Tables

It is time to establish our table structure by identifying subjects for our fields list. To complete this activity, we will locate subjects implied by our fields list.

Why did we start with the field list and not the lists of subjects? If you remember how we defined a field earlier, “A field helps organize and store one specific type of information about each item in a database,”. Fields offer an unbiased perspective compared to a list of subjects.

As we have refined our characteristic list to become a complete list of fields, it is best practice to have a predefined table list from the start. The Predefined table list exists from our data collection analysis and information-gathering phase of the interviews.

Step 1: Using the Preliminary Field List to identify subjects

We will uniquely identify subjects by allowing our field list to expressly suggest subjects that will end up being tables in the database. After that, we will verify and cross-check these subjects with our predefined table list.

For example, let's assume the information below is our predefined table list following the outcome of the interviews and the data collection process that was launched earlier:

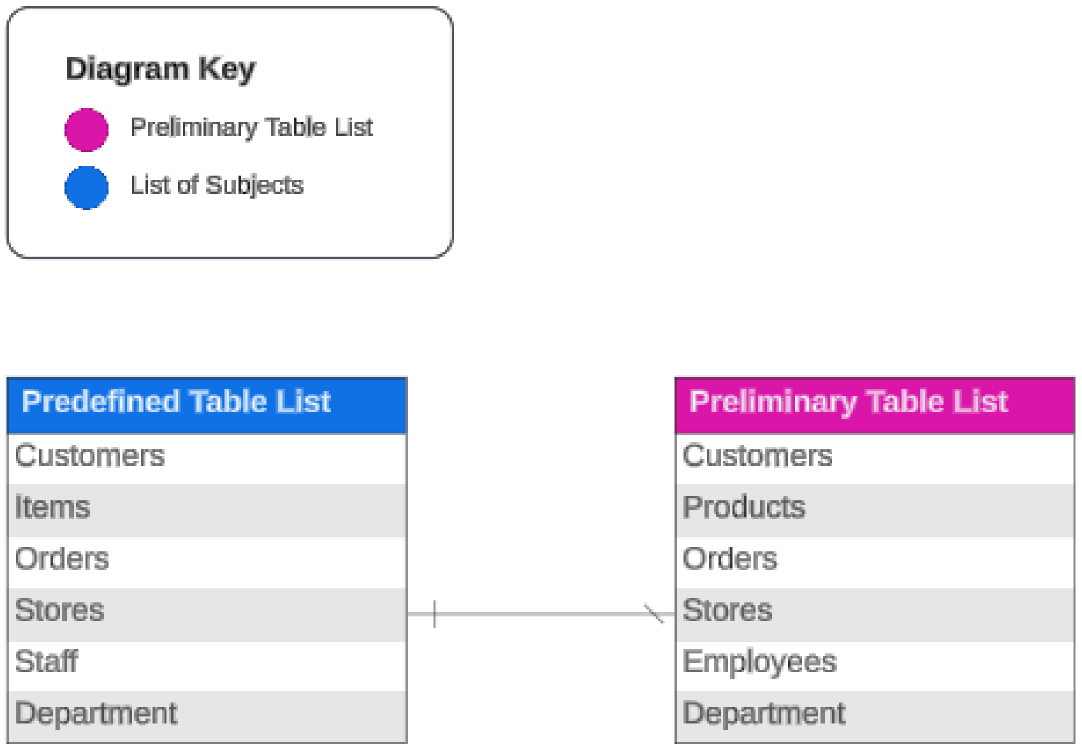
1. Customers
2. Items
3. Orders
4. Stores
5. Staff
6. Department

Now, we have the first (1) version of our suggested preliminary Table list, with the help of the Preliminary field list we identified earlier:

1. Customers
2. Products
3. Orders
4. Stores
5. Employees
6. Departments

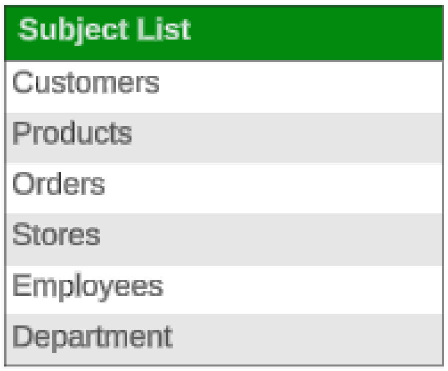
Step 2: Merging Preliminary Table List and Predefined Table List

The next step in finalizing the subject list (Preliminary Table List 1) is to merge the tables from the preliminary field list and the predefined table list.



The list of Subjects and the Preliminary Table List for Crocs, the company.

The final version of our merged Table list is below == called **Preliminary Subject List 1**



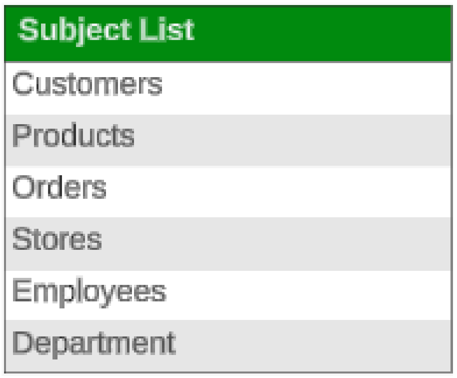
Subject List
Customers
Products
Orders
Stores
Employees
Department

Step 3—Final Step: Using Crocs's goals and mission objectives (the client) to cross-check and validate our subjects.

This step allows us to review and refine Preliminary Table List 1 above and cross-check if they align with each Croc's mission objective item. Here, the suggested Table list above properly aligns with the goals and objectives identified for our database.

Pro Tip: In your future database design, it is best practice to thoroughly iterate through your mission objectives to determine whether to accommodate changes to the Subject list. These changes could be an addition of a new list or a removal, as the case may be.

Our Final Table List



Subject List
Customers
Products
Orders
Stores
Employees
Department

The table below describes our Subject list by introducing the table types and their descriptions.

Name	Type	Description
Customers	Data	The Customer Table in the Crocs database stores key details about customers, including customer ID, name, contact information, and addresses, supporting order processing and customer management.
Products	Data	The Product Table in the Crocs database stores essential product information, including product ID, name, description, category, price, and stock levels, facilitating inventory and sales management.
Orders	Data	The Orders Table in the Crocs database stores critical order details, including order ID, customer ID, product IDs, quantities, order date, shipping date, and status, supporting order tracking and fulfillment.
Stores	Data	The Stores Table in the Crocs database stores vital store information, including store ID, name, location, contact details, and employee ID, aiding in store management and operations.
OrderDetails	Data	The OrderDetails Table in the Crocs database stores specific order line information, including order ID, product ID, quantity, price, and discount, facilitating detailed order processing and inventory management.
Employees	Data	The Employees Table in the Crocs database stores essential employee information, including employee ID, name, department, contact details, etc, supporting human resources and staff management.
Department	Data	The Department Table in the Crocs database stores key department information, including department ID, name, and manager ID, facilitating organizational structure and department management.

Establishing Table Relationships

Keys are mainly used to establish and enforce data integrity and relationships between tables. They ensure that every record in a table is uniquely identified.

There are four main types of keys. However, our main focus in the database is primary and foreign keys. You can read up on what these keys represent in the terminologies section of this article

1. Candidate Key
2. Primary Key
3. Foreign Key
4. Non Keys.

Step 1: Specifying primary keys

Now that we have created our table list, each table needs to include a column or set of columns uniquely identifying each row stored in the table. In most cases, it is called a unique identification number (ID) in our tables, such as a Customer ID number or a Product ID number. In a database concept, this information is called the primary key of the table.

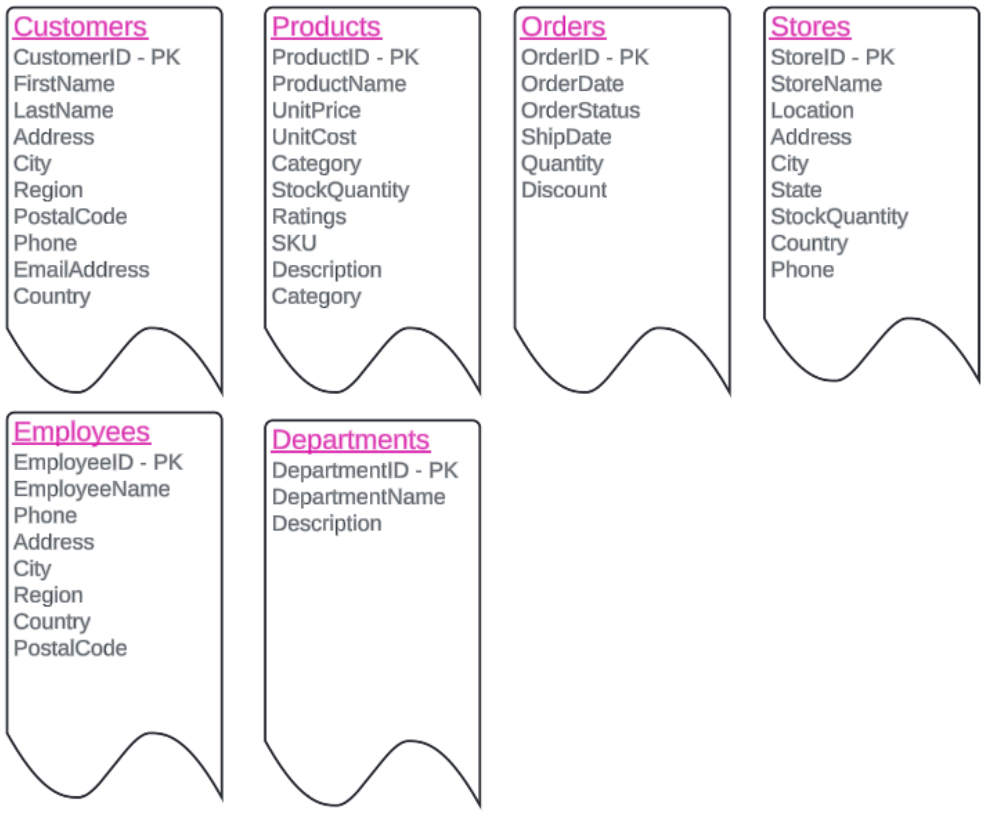
You can use an existing unique identifier, such as a product number that uniquely identifies every item in your catalog, as the primary key for the table if it is already established, but only if the values in this column will vary for every record. Primary Keys have to remain unique. For example, don't use people's names as a primary key because names are not unique. You could easily have two people with the same name on the same table.

A primary key additionally needs to have a value at all times. If a column's value can become unassigned or unknown (a missing value) at some point, it can't be used as a component in a primary key. Hence the need to maintain a unique primary key that will rarely be edited in the future.

For our database, we introduced a new primary key uniquely identifying each record in our tables.

In some cases, you may want to use two or more fields that, together, provide the primary key of a table. For example, our Order table that stores Crocs items for orders would use two columns in its primary key: Order ID and Product ID. When a primary key employs more than one column, it is also called a composite key.

For our Crocs (the company) database, we created an AutoNumber column for each of the tables to serve as the primary key: CustomerID for the Customers table, ProductID for the Products table, OrderID for the Orders table, StoreID for the Stores table, ReviewID for the Reviews table, EmployeeID for the Employees table, and DepartmentID for the Departments table.



It is time to connect all of the tables together now that we have the necessary data, including the primary keys for every table. The purpose of this is to help bring the information together in meaningful ways.

Imagine for a second that you wish to view the number of products a customer ordered this month against their last 4 months. It is impossible to view the entire information without creating table relationships. The tables will remain alone and will never be able to provide any meaningful information.

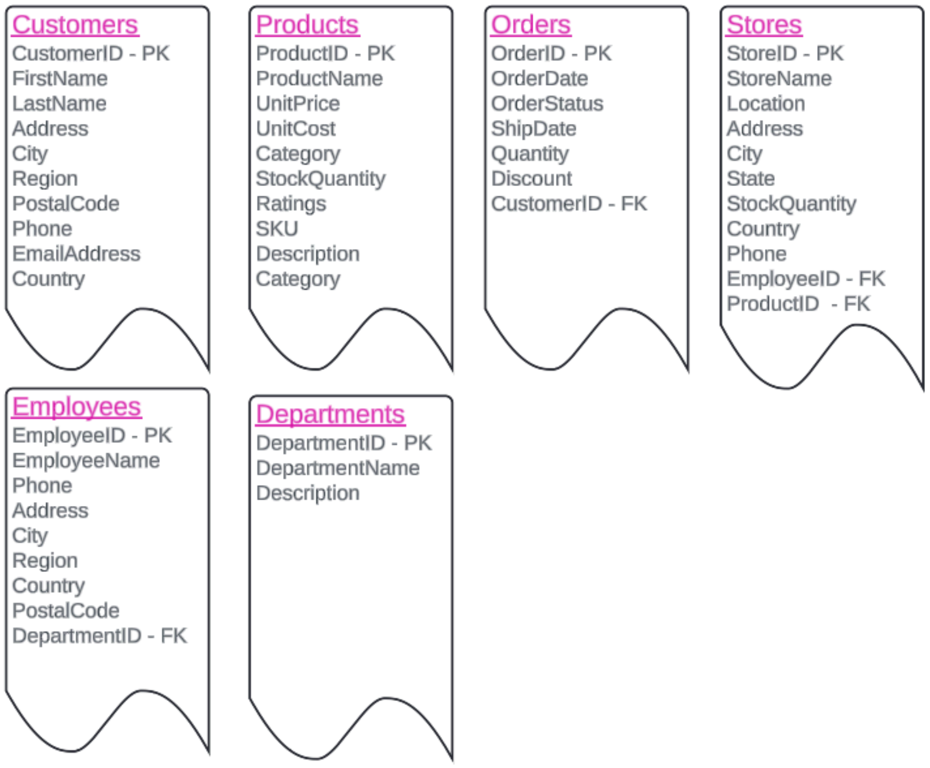
Step 2: Creating one-to-many relationships:

Consider this example: the Customers and Orders tables in our Crocs database. A customer can take as many others. It follows that for any customer represented in the Customers table, there can be many orders represented in the Orders table. The relationship between the Customers table and the Orders table is, therefore, a one-to-many relationship.

To represent a one-to-many relationship in our database design, we took the primary key on the "one" side of the relationship and added it as an additional column (s) to the table on the "many" side of the relationship. In this case, we added the Customer ID column from the Customers table to the Orders table. We can then use the Customer ID number in the Orders table to locate or trace the customer who placed such orders.

The Customer ID column in the Orders table is called a foreign key. A foreign key is another table’s primary key. The Customer ID column in the Orders table is a foreign key because it is also the primary key in the Customers table. We also applied this step to each of our stores and employees' tables. We added the Product ID column from the Products table to the Stores table and the Department ID column from the Departments table to the Employees table.

It is also best practice to always form a basis for joining related tables by establishing pairings of primary keys and foreign keys. Establishing one-to-many relationships helps you truly determine that the two tables involved will indeed require a shared column.



Step 3: Creating a many-to-many relationships:

Consider the relationship between the Products table and the Orders table.

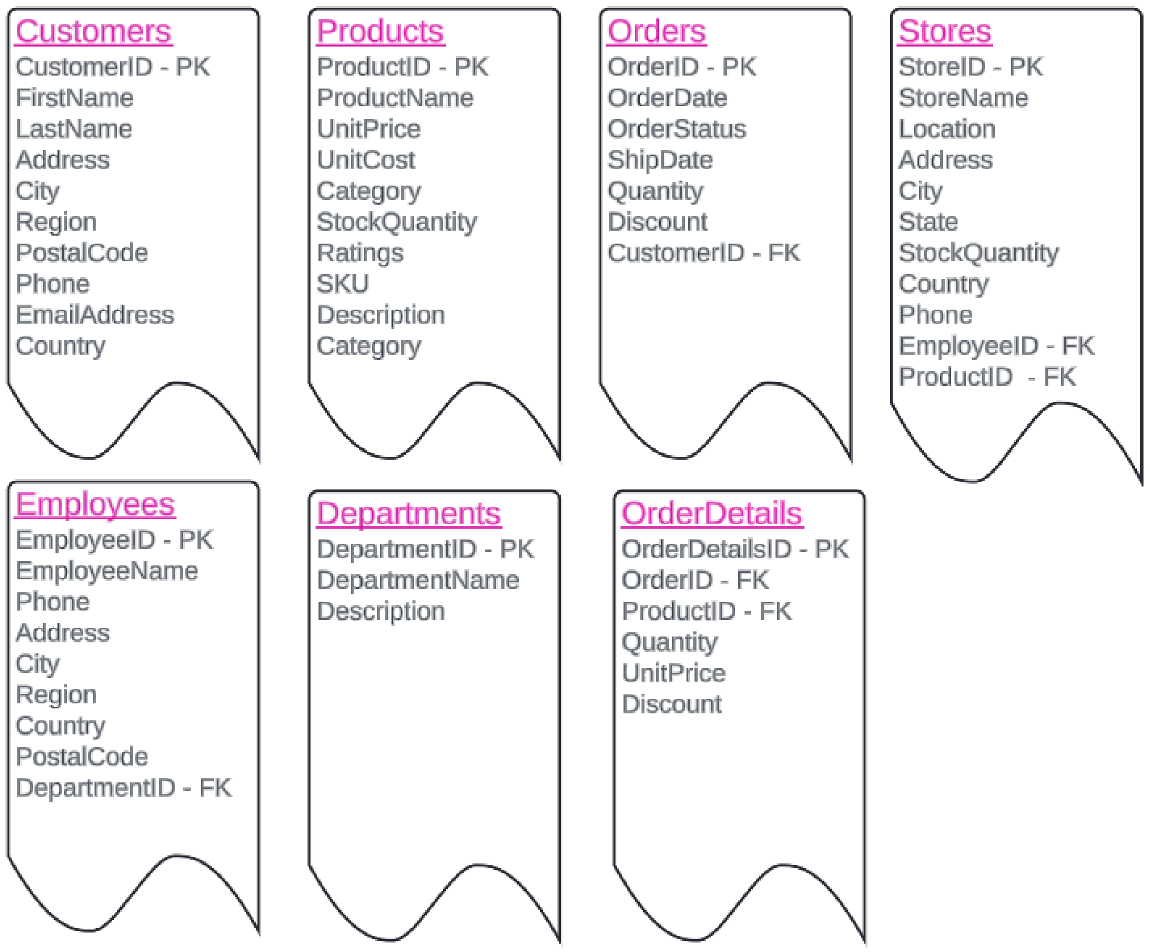
An order can contain multiple products, and a product can be included in multiple orders. This means each entry in the Orders table may correspond to several entries in the Products table, and vice versa. This is known as a many-to-many relationship because a product can be part of many orders, and an order can include many products. To identify many-to-many relationships in your tables, examining both sides of the relationship is essential.

The orders and products tables have a many-to-many relationship, which can cause complications. For example, if you add the Product ID to the Orders table, you'd need multiple records for each order to include different products. This would duplicate order information, making the design inefficient and prone to errors. Similarly, adding the Order ID to the Products table would mean having multiple records for each product.

To solve this, you can create a third table, known as a junction table. This table breaks the many-to-many relationship into two one-to-many relationships by including the primary keys from both the Orders and Products tables. This way, the junction table records each instance of the relationship.

Each record in the Order Details table represents a single line item in an order. The primary key for this table comprises two fields: the foreign keys from the Orders and Products tables. Using only the Order ID wouldn't work as the primary key because an order can have multiple line items, meaning the Order ID would be repeated. Similarly, using only the Product ID wouldn't work because a product can be part of many different orders. However, combining both fields ensures each record is unique.

The Orders and Products tables are not directly linked in our Crocs database. Instead, they are connected through the Order Details table. This setup turns the many-to-many relationship between orders and products into two one-to-many relationships.



With the addition of the Order Details table, the list of tables and their fields now appears as follows:

Reviewing and refining the Design

Now that we have the tables, fields, and relationships we need, we populated our tables with some of the existing data, which was collated during the data collection and analysis stage. We tried working with this information by creating queries and views, adding new records, and so on. Doing this helps us highlight potential problems—for example, if we need to add a column that we forgot to insert during the design phase or if a table needs to be split into two tables to remove duplicates.

This also allowed us to create a few reports and analyze if our database could get us the answers we wanted. The review and refinement phase is very critical, as you will probably discover room for improvement.

Defining Business Rules for the Database

We are heading to the end of our database design. Creating and defining business rules for the database involves establishing guidelines that enforce business policies and procedures within the database structure. These rules ensure data integrity, consistency, and adherence to business objectives for building the database, as well as simplify operations.

Here are the five business rules that have been created and executed within the Crocs database:

1. **Unique Product SKUs**
Business Rule: Each product must have a unique Stock Keeping Unit (SKU).

Implementation:
A unique constraint has been added to the SKU field in the Products table.

2. **Valid Price Range for Products**
Business Rule: The price of any product must be greater than $0 and less than $1000.

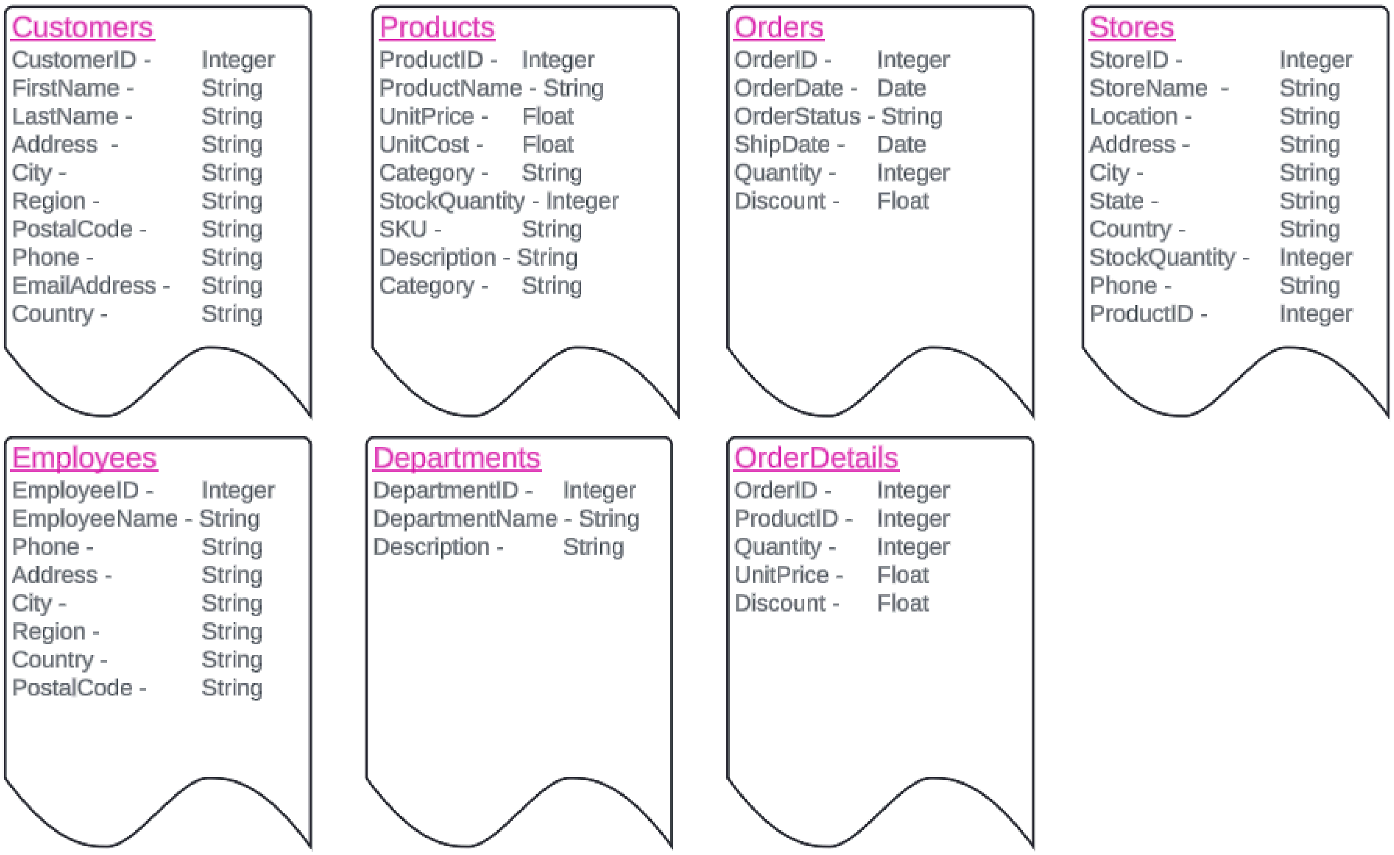
Implementation:
We added a check constraint to the Price field in the Products table.

3. **Mandatory Customer Email**
Business Rule: Every customer must have an email address on record.

Implementation:
We ensured the Customers table's email field was set to NOT NULL.

4. **Order Status Workflow**
Business Rule: An order can only transition from 'Pending' to 'Shipped' status and then to 'Delivered'. It cannot skip statuses or revert to 'Pending' once shipped.

Implementation:
We introduced triggers, or application logic, to enforce this workflow.



Customers

CustomerID -	Integer
FirstName -	String
LastName -	String
Address -	String
City -	String
Region -	String
PostalCode -	String
Phone -	String
EmailAddress -	String
Country -	String

Products

ProductID -	Integer
ProductName -	String
UnitPrice -	Float
UnitCost -	Float
Category -	String
StockQuantity -	Integer
SKU -	String
Description -	String
Category -	String

Orders

OrderID -	Integer
OrderDate -	Date
OrderStatus -	String
ShipDate -	Date
Quantity -	Integer
Discount -	Float

Stores

StoreID -	Integer
StoreName -	String
Location -	String
Address -	String
City -	String
State -	String
Country -	String
StockQuantity -	Integer
Phone -	String
ProductID -	Integer

Employees

EmployeeID -	Integer
EmployeeName -	String
Phone -	String
Address -	String
City -	String
Region -	String
Country -	String
PostalCode -	String

Departments

DepartmentID -	Integer
DepartmentName -	String
Description -	String

OrderDetails

OrderID -	Integer
ProductID -	Integer
Quantity -	Integer
UnitPrice -	Float
Discount -	Float

Again, data scientists don't define the business rules or design the database based on what they think is right. The users and management define the rules—the owners of the database. We collaborate with users and management to ensure meaningful and clear constraints. As the consultant, you have the freedom to schedule meetings for group discussions to define and establish appropriate business rules.

Conclusion

In this article, we've walked through the entire process of designing a database from start to finish, using Crocs as our example. By understanding and applying good design principles, we've created a database that is robust, scalable, and aligned with business objectives. From initial interviews to defining business rules, each step is crucial to ensuring the database meets the needs of the client and can adapt to future changes.

Key takeaways include the importance of maintaining data integrity, organizing data effectively, and establishing clear relationships between tabldes. By following these guidelines, you can create a database that not only stores information efficiently but also enhances business operations and decision-making processes.

Remember, a well-designed database is a powerful tool that supports business growth and efficiency. By investing the time and effort into proper database design, you ensure that your data management processes are streamlined, accurate, and ready to support your business needs both now and in the future.