**E-Commerce Database Requirements**

**Goals & Overview**

* To understand the requirements given by the stakeholders.
* To identify what entities are required and the attributes corresponding to them.
* To plan on which relational database management to develop e-Commerce project.
* To design the database with the core entities and establish the relations between them.
* To ensure that the data stored is accurate and consistent by implementing validation rules and quality checks.
* To implement the customer queries to work efficiently with the DDL and DML commands
* To document the result in the form reports by implementing tools for data mining and data visualization.
* Overall goal of database project is to create a system that meets the specific needs of the users and help them to manage and make use of their data more effectively.

The steps to design and implement a database are as follows

1. Requirement Specification & Analysis:

This phase involves gathering information from stakeholders and analyzing their needs. Which includes functional and nonfunctional requirements for the database system.

The business here is an ecommerce website and where product & customer information is stored and dummy orders are created to understand database concepts.

The database has to store Customer Information, Product details they have in the inventory, Order details, Payment & Shipping information.

After the analysis, entities that are identified based on the requirements are

CUSTOMER, OREDER, PRODUCT, SHIPMENT.

1. Conceptual Design:

The next step is to create a conceptual schema for the database, using a high-level conceptual data model. The result of this phase is an Entity-Relationships (ER) diagram or UML class diagram. It is a high-level data model of the application.It describes how different entities (object,items) are related to each other.It also describes what attributes (features) each entity has.It includes the relationship between entities, attributes of the application.

**ENTITIES:**

* CUSTOMER – customer\_id, first\_name, last\_name, email, address, city, state, zip
* ORDER -- order\_id, order\_date, order\_cost, order\_status
* PRODUCT -- product\_id, product\_name, product\_description, price, quantity
* SHIPMENT -- shipment\_id, ship\_date, courier\_name, ship\_status

**CUSTOMER** : This entity will store the information about the customers who shop at the online store. We declare “customer\_id” to be primary key that uniquely identifies every record of the table. The attribute “email” can’t be a primary key since it takes more space and more time for processing large text than an integer datatype.

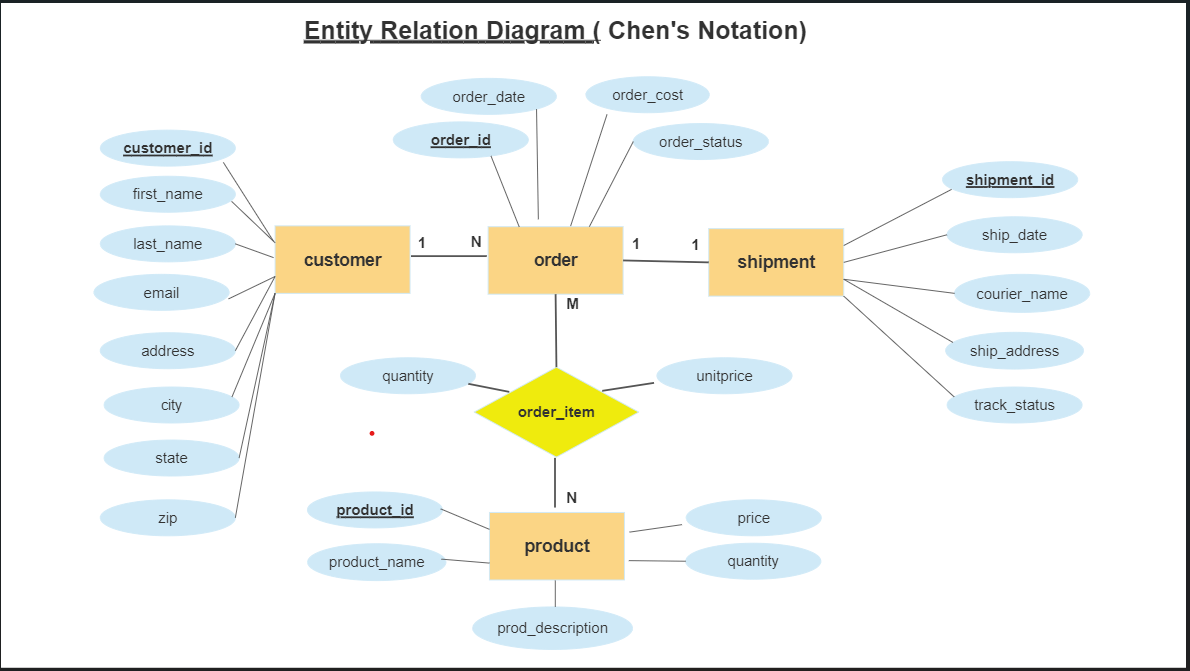
As index been created on primary key by default. Most RDBMSs build a clustered index on the primary key to facilitate fast search and retrieval. The other attributes in this entity are customer name, address, city, state and zip.

**ORDER**: This entity will store information about the orders placed by the customers. We declare “order\_id” to be primary key. The other attributes are the date of the order , cost of the order and the status of order pertains to ‘confirmed’/’canceled’.

Order entity will have

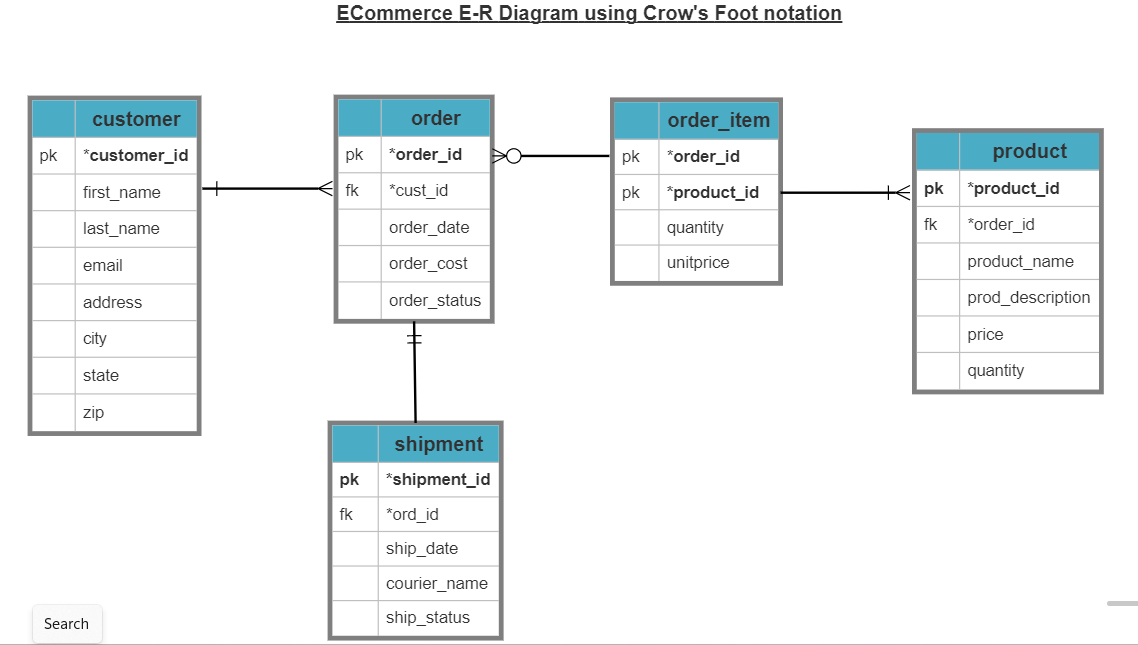
**PRODUCT:** This entity will store information about the products that the online store sells. We declare “product\_id” to be primary key. The other attributes are name of the product, description of the product , product price and quantity.

**SHIPMENT:** This entity will store information about the shipment of orders such as “shipment\_id” to be primary key, shipment date ,courier name and status of shipment (delivered/intransit).

****

**3.** Logical Design

The result of the logical design phase (or data model mapping phase) is a set of relational schemas. The ER diagram or class diagram is the basis for these. In this phase ,the primary keys and foreign keys are defined.

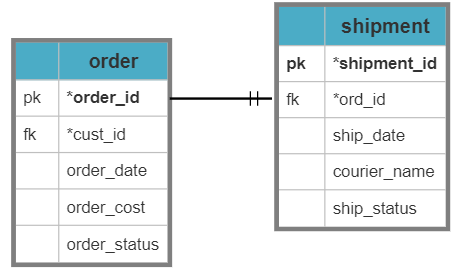


The most crucial aspect in designing a relational database is to identify the relationships among tables. The types of relationship include:

* one-to-one
* one-to-many
* many-to-many

one-to-one: In database one-to-one relationship is a type of relationship between 2 entities or tables where one record in the first table is associated with exactly one record in the second table, and vice versa. Typically, one table will have a primary key that uniquely identifies each record , and the other table will have a foreign key that references the primary key of the first table.

In our E-commerce application, we assume that an order will be shipped to one customer’s address. The two entities Order and Shipment exhibit a one-to-one relationship. That is for every row in parent table ,there is one corresponding row in the child table. The **order\_id** attribute of order entity is a **primary key** and it becomes the **foreign key** column in the **shipment** entity.



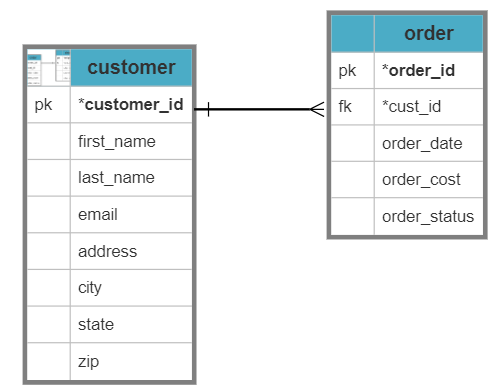
**One-to-many:**

In database design,a one-to-many relationship is a type of relationship between two entities or tables where one record in the first table is associated with zero or more records in the second table, but each record in the second table is associated with exactly one record in the first table.

One-to many relatioships are widely used in database design,as they allow for efficient organization and management of related data.They enable the representation of complex relationships and provide flexibility in querying and retrieving data.

In e-commerce database, the **customer** table may have **one-to-many** relationship with **order** table. Where each order belongs to only one customer and a customer can place one or multiple orders. The **customer** entity has **customer\_id** as **primary key** and the same **customer\_id** acts as **foreign\_key** in **order** table.

The **order** table may have **one-to-many** relationship with **order-item** table. Each order can have multiple items, but each item belongs to only order.



**many-to-many:**

A many-to-many relationship is a type of relationship between two entities or tables where multiple records in the first table are associated with multiple records in the second table, and vice versa. It means that each record in one table can be related to multiple records in other table, and vice versa.

The relationship between **order** entity and **product** entity is a many-to-many relationship.