**Understand normalization**

Completed100 XP

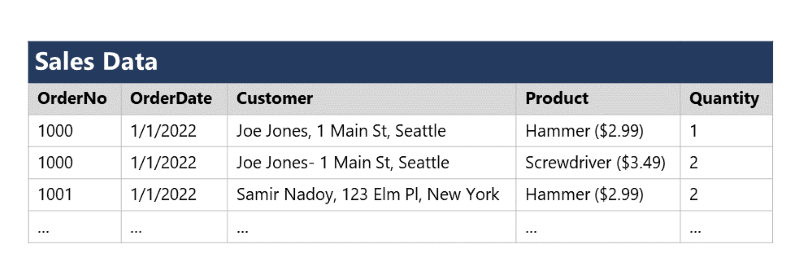
* 6 minutes

Normalization is a term used by database professionals for a schema design process that minimizes data duplication and enforces data integrity.

While there are many complex rules that define the process of refactoring data into various levels (or *forms*) of normalization, a simple definition for practical purposes is:

1. Separate each *entity* into its own table.
2. Separate each discrete *attribute* into its own column.
3. Uniquely identify each entity instance (row) using a *primary key*.
4. Use *foreign key* columns to link related entities.

To understand the core principles of normalization, suppose the following table represents a spreadsheet that a company uses to track its sales.



Notice that the customer and product details are duplicated for each individual item sold; and that the customer name and postal address, and the product name and price are combined in the same spreadsheet cells.

Now let's look at how normalization changes the way the data is stored.



Each entity that is represented in the data (customer, product, sales order, and line item) is stored in its own table, and each discrete attribute of those entities is in its own column.

Recording each instance of an entity as a row in an entity-specific table removes duplication of data. For example, to change a customer's address, you need only modify the value in a single row.

The decomposition of attributes into individual columns ensures that each value is constrained to an appropriate data type - for example, product prices must be decimal values, while line item quantities must be integer numbers. Additionally, the creation of individual columns provides a useful level of granularity in the data for querying - for example, you can easily filter customers to those who live in a specific city.

Instances of each entity are uniquely identified by an ID or other key value, known as a *primary key*; and when one entity references another (for example, an order has an associated customer), the primary key of the related entity is stored as a *foreign key*. You can look up the address of the customer (which is stored only once) for each record in the **Order** table by referencing the corresponding record in the **Customer** table. Typically, a relational database management system (RDBMS) can enforce referential integrity to ensure that a value entered into a foreign key field has an existing corresponding primary key in the related table – for example, preventing orders for non-existent customers.

In some cases, a key (primary or foreign) can be defined as a *composite* key based on a unique combination of multiple columns. For example, the **LineItem** table in the example above uses a unique combination of **OrderNo** and **ItemNo** to identify a line item from an individual order.

**Chuẩn hóa dữ liệu** là thuật ngữ được các chuyên gia về cơ sở dữ liệu sử dụng cho quy trình thiết kế lược đồ nhằm giảm thiểu việc trùng lặp dữ liệu và đảm bảo tính toàn vẹn của dữ liệu.

Mặc dù có nhiều quy tắc phức tạp xác định quá trình tái cấu trúc dữ liệu thành các cấp độ (hoặc dạng) chuẩn hóa khác nhau, nhưng định nghĩa đơn giản cho các mục đích thực tế là:

1. **Phân tách từng thực thể riêng thành một bảng riêng.**
2. **Phân tách từng thuộc tính riêng biệt thành một cột riêng.**
3. **Xác định duy nhất từng thể hiện thực thể (hàng) bằng một khóa chính.**
4. **Sử dụng các cột khóa ngoại để liên kết các thực thể liên quan.**

Để hiểu các nguyên tắc cốt lõi của chuẩn hóa dữ liệu, giả sử bảng sau đây đại diện cho một bảng tính mà công ty sử dụng để theo dõi doanh số bán hàng của mình.

Chú ý rằng thông tin chi tiết về khách hàng và sản phẩm bị trùng lặp cho mỗi mặt hàng được bán riêng lẻ; và tên khách hàng và địa chỉ bưu điện, tên sản phẩm và giá được kết hợp trong cùng một ô bảng tính.

Bây giờ, chúng ta hãy xem cách chuẩn hóa thay đổi cách lưu trữ dữ liệu

**Mỗi thực thể được biểu diễn trong dữ liệu (khách hàng, sản phẩm, đơn đặt hàng và mặt hàng) được lưu trữ trong một bảng riêng và mỗi thuộc tính riêng biệt của các thực thể đó nằm trong một cột riêng.**

Việc ghi lại từng thể hiện của một thực thể dưới dạng một hàng trong bảng dành riêng cho thực thể đó sẽ loại bỏ sự trùng lặp dữ liệu. Ví dụ: để thay đổi địa chỉ của khách hàng, bạn chỉ cần sửa đổi giá trị trong một hàng duy nhất.

**Việc phân tách các thuộc tính thành các cột riêng lẻ đảm bảo rằng mỗi giá trị được hạn chế theo một kiểu dữ liệu thích hợp** - ví dụ: giá sản phẩm phải là giá trị thập phân, trong khi số lượng mặt hàng phải là số nguyên. Ngoài ra, việc tạo các cột riêng lẻ cung cấp mức độ chi tiết hữu ích trong dữ liệu để truy vấn - ví dụ, bạn có thể dễ dàng lọc khách hàng theo những người sống trong một thành phố cụ thể.

**Các thể hiện của mỗi thực thể được xác định duy nhất bằng ID hoặc giá trị khóa khác, được gọi là khóa chính.** Khi một thực thể tham chiếu đến thực thể khác (ví dụ: một đơn hàng có một khách hàng được liên kết), khóa chính của thực thể liên quan được lưu trữ dưới dạng khóa ngoại. Bạn có thể tra cứu địa chỉ của khách hàng (chỉ được lưu trữ một lần) cho mỗi bản ghi trong bảng Đơn hàng bằng cách tham chiếu đến bản ghi tương ứng trong bảng Khách hàng. Thông thường, một hệ thống quản lý cơ sở dữ liệu quan hệ (RDBMS) có thể thực thi tính toàn vẹn tham chiếu để đảm bảo rằng giá trị được nhập vào trường khóa ngoại có một khóa chính tương ứng tồn tại trong bảng liên quan - ví dụ: ngăn chặn các đơn đặt hàng cho những khách hàng không tồn tại.

**Trong một số trường hợp, khóa (chính hoặc ngoại) có thể được định nghĩa là khóa tổng hợp dựa trên sự kết hợp duy nhất của nhiều cột.** Ví dụ: bảng Mặt hàng (LineItem) trong ví dụ trên sử dụng sự kết hợp duy nhất của Số đơn hàng (OrderNo) và Số mặt hàng (ItemNo) để xác định một mặt hàng từ một đơn hàng riêng lẻ.