"Database Introduction"

book "Data Wrangling with SQL"

Introduction

- **Data-driven decision-making is essential in today's fast-paced world.**
- **Databases play a crucial role in organizing and managing large volumes of data.**
- **Understanding databases and SQL is vital for effective data wrangling.**

Chapter 1: Getting Started

1.1 Efficient Data Organization

- **Databases provide structured data storage and retrieval.**
- **Learning database design, data modeling, and normalization is key.**
- **Optimized database structures ensure smooth data-wrangling.**
-

1.2 Data Integrity and Consistency

- **Data integrity and consistency are vital in data wrangling.**
- **Databases use constraints and relationships to maintain data quality.**
- **A clear understanding of data consistency ensures reliable insights.**

"Decoding Database Structures – Relational and Non-relational"

Introduction

- **Understanding the concept of a database is crucial before delving into relational and non-relational databases.**
- **A database is an organized collection of information used for tracking and quick information retrieval.**

What is a Database?

- **A database helps in logically organizing information.**
- **Using logical categorization, information can be located quickly.**
- **In a superstore, information is categorized into sections, rows, and columns, making it accessible through a database.**
- **A database is a collection of information stored on a server for regular analysis and decision-making.**
- **Data is organized into tables, similar to spreadsheets, containing rows and columns.**
- **Tables can be used for various categories or clients (e.g., university, superstore).**
- **Each row represents a specific occurrence or transaction.**

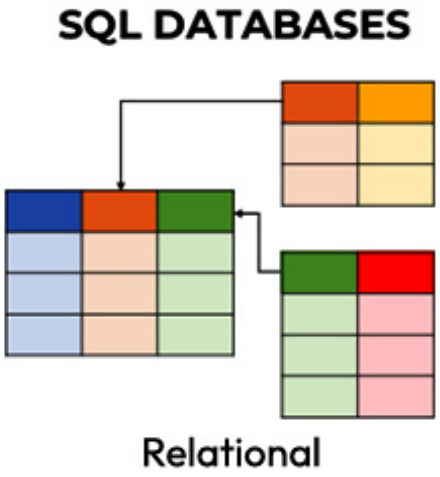
Types of Databases

- **DBMS**

- **Database Management Systems (DBMSs) are used to store and manage data.**
- **SQL is the common language for extracting data.**
- **Databases can be relational or non-relational.**
- **Relational Databases**
- **Store data in tables or entities.**
- **Data about entities is stored in relations (2D tables with rows and columns).**
- **Each row contains data, each column contains attributes.**
- **Tables have a primary key for unique identification.**
- **Relationships between fields and tables are defined as schemas.**
- **SQL queries are used to extract data.**

- ***Advantages of Relational Databases:***

- **Highly efficient**
- **High readability due to sorted and unique data**
- **High data integrity**
- **Normalized data**

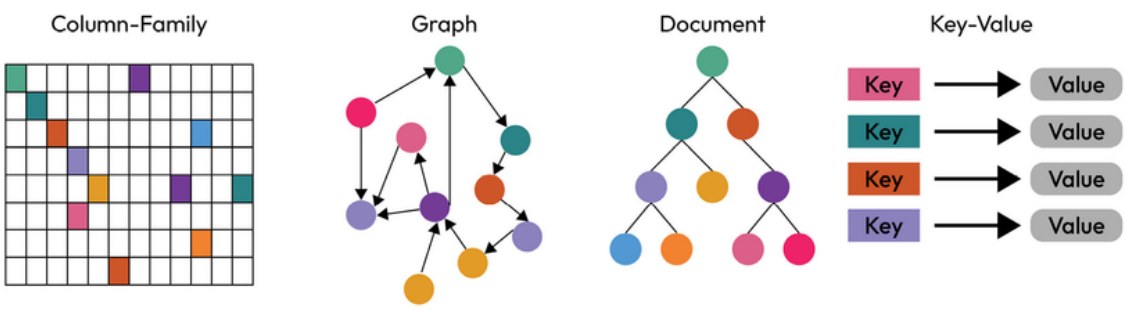


- **Non-relational Databases (NoSQL)**

- **Store data in non-tabular formats (e.g., key-value, document-based).**
- **Key-value databases use unique keys to connect to values.**
- **Document-oriented databases use complex combinations of key-value pairs.**
- **Non-relational databases are more flexible.**
- **Common NoSQL databases: MongoDB, Cassandra, Amazon DynamoDB, Apache HBase.**

- ***Advantages of Non-relational Databases:***

- **Simple management, no sorting needed**
- **Higher readability of specific documents, especially for big data**
- **Scalable with server splitting into multiple clusters**
- **Multiple clusters allow for more powerful and scalable computing.**
- **Distributing workloads efficiently and achieving better performance.**
- **Advantages include scalability, improved performance, fault tolerance, and efficient handling of large workloads.**



Key	Document
2022	{ Customer ID: 1234, Customer Name: Joey, Customer address: XYZTX, Order details: { Order 1: {product 1, product description} Order 2: {Product 1, product description} }}

"Tables and Relationships"

Tables and Relationships

- **In database management, tables and relationships are fundamental for organizing and connecting data in SQL Server.**
- **SQL Server empowers businesses and organizations to efficiently store, retrieve, and analyze vast amounts of information.**

The SQL CREATE DATABASE Statement

- **All tables in SQL Server are stored in a repository known as a database.**
- **Databases are collections of tables related to specific entities (e.g., insurance companies).**
- **A database administrator owns the database and manages access permissions.**
- **The syntax for creating a database is**

  CREATE DATABASE Database_name.
- **Example: CREATE DATABASE Superstore_DB.**
- **The syntax for deleting a database is**

  DROP DATABASE Database_name.
- **Example: DROP DATABASE Superstore_DB.**
- **The syntax for viewing the entire database is**

  SHOW DATABASE.
- **Example: SHOW DATABASE.**

The SQL CREATE TABLE Statement

- **The CREATE TABLE statement is used to create a new table, consisting of rows and columns, in a specified database.**
- **Example: Creating a table named walmart.customer_info with columns like Customer_ID, Name, Address, Email, and Phone.**

```
CREATE TABLE DATABASENAME.TABLE_NAME(
Column 1 datatype,
Column 2 datatype,
Column 3 datatype,
);
```

For example. we can create a **walmart.customer_info** table as follows:

```
(Customer_ID int,
Name varchar(255),
Address varchar(255),
Email varchar(255),
Phone varchar(255)
)
```

- **Columns have specific data types, e.g., varchar for text and int for integers.**
- **A table is created with defined columns and their data types.**

SQL DROP TABLE versus TRUNCATE TABLE

- **To delete an entire table with its schema, use DROP TABLE table_name.**
- **Example: DROP TABLE Walmart.customer_info.**
- **To delete only the table's content while retaining the structure, use TRUNCATE TABLE table_name.**
- **Example: TRUNCATE TABLE Walmart.customer_info.**
- **After truncation, the table structure remains, but it has zero rows.**

SQL ALTER TABLE

- **The ALTER TABLE SQL syntax is used to insert, update, or delete data or columns in a pre-created table.**
- **Example: ALTER TABLE Walmart.customer_info DROP COLUMN Email.**
- **This statement drops the "Email" column from the table.**

SQL Constraints

- **Constraints are predefined rules set by a database administrator or table creator to ensure data uniqueness and cleanliness.**
- **Constraints can be defined at the table or column level.**

Common constraints include:

- ***The UNIQUE constraint*, ensuring all columns have unique values.**
- ***The NOT NULL constraint*, ensuring all columns have values.**
- ***PRIMARY KEY*, a unique value at each row level.**
- ***FOREIGN KEY*, a relational key connecting tables within the same database.**

SQL Keys

- **In relational databases (RDBMSes), keys establish relationships between tables, enabling data retrieval.**
- **Types of keys include primary keys, foreign keys, and candidate keys.**

• Candidate Keys

- **A candidate key is a set of one or more columns uniquely identifying a record in a table.**
- **It can serve as a primary key as it cannot be null and must be unique.**
- **A candidate key is a super key with no repeating attributes.**
- **Out of all possible candidate keys, only one is selected as the primary key.**
- **In the example, "Passport_Number" is a candidate key, and "Customer_ID" is a primary key.**

Example:

Consider a table "Customer" with the following data:

Customer_ID	1	2	3
Name	Adam	James	Paul
Passport_Number	L08790	L08791	L08792
DOB	7/11/1990	8/6/1992	3/4/1993

In this table, "Passport_Number" is a candidate key, and "Customer_ID" is the primary key. This means "Passport_Number" is unique for each customer, and "Customer_ID" uniquely identifies each customer.

- ## Primary Keys

- **A primary key is an attribute used to uniquely identify a row in a table.**
- **It ensures uniqueness and is essential for data integrity.**
- **In the table, Customer_ID would be the primary key, while Passport_Number would not be a primary key due to containing confidential information.**

Example:

Consider a table "Customer" with the following data:

Customer_ID	1	2	3	→ primary key
Name	Adam	James	Paul	
Passport_Number	L08790	L08791	L08792	
DOB	7/11/1990	8/6/1992	3/4/1993	

In this table, "Customer_ID" is the primary key as it uniquely identifies each customer.

- ## Alternate Keys

- **An alternate key is a candidate key that hasn't been assigned as the primary key.**
- **It can also uniquely identify a row.**
- **In the table, "License_Number" can be an alternate key as it can also uniquely identify a customer.**

Example:

Customer_ID	1	2	3	→ candidate key
Name	Adam	James	Paul	
Passport_Number	L08790	L08791	L08792	
DOB	7/11/1990	8/6/1992	3/4/1993	
License_Number	L01YZ	L02ER	L03PX	

In this table, "Customer_ID" is the primary key, and "License_Number" is an alternate key.

- ## Super Keys

- **A super key is formed when more than one attribute can be used to uniquely identify a row.**
- **It is a broader concept than a primary key.**
- **For example, "Customer_ID + Name" can be a super key, as it ensures uniqueness when combined.**

Example:

Customer_ID	1	2	3	→ Super key
Name	Adam	James	Paul	
Passport_Number	L08790	L08791	L08792	
DOB	7/11/1990	8/6/1992	3/4/1993	
License_Number	L01YZ	L02ER	L03PX	

In this table, "Customer_ID + Name" is a super key, as it ensures uniqueness when combined.

- ## Composite Keys

- **When a table lacks an individual attribute that can serve as a candidate key, a composite key is used.**
- **It consists of two or more columns to create a unique key.**
- **In this scenario, duplicate rows are possible if both columns have the same values.**

Example:

Customer_ID	1	2	3	}
Name	Adam	James	Paul	→ Composite key
Passport_Number	L08790	L08791	L08792	
DOB	7/11/1990	8/6/1992	3/4/1993	

A composite key, such as "Name + DOB," may be used if neither "Customer_ID" nor "Passport_Number" can uniquely identify a row.

Surrogate Keys

- **A surrogate key is generated by the system and has no business meaning.**
- **It is typically used when there is no suitable primary key.**
- **The values are sequential and serve as a primary key.**
- **In the "Address" table, "Address_ID" is a surrogate key.**

Example:

Address_ID	Street_Name	City	State	Zipcode
1	Jefferson St	Dallas	Texas	38256
2	Thomas St	Memphis	Tennessee	38257
3	James St	Chicago	Illinois	33189
4	Perkins St	Miami	Florida	23487

In this table, "Address_ID" is a surrogate key, sequentially generated, and holds no business value.

Primary Keys in Detail

- **A primary key uniquely identifies each record in a table.**
- **It holds unique values for each record and cannot have null values.**
- **Primary keys and foreign keys are used to establish relationships between tables in a relational database.**
- **Examples include Social Security Number (SSN) and passport numbers.**

Foreign Keys in Detail

- **A foreign key links data in one table (referencing table) with another table containing primary key values (referenced table).**
- **It creates cross-references between tables, maintaining referential integrity.**
- **The referenced table with the primary key is called the parent table, and the referencing table with the foreign key is the child table.**
- **A foreign key ensures that values in the referencing table exist in the referenced table, preventing data inconsistencies.**

Primary_Key

Customer_ID	Name	Address
1	Joey	Texas
2	Ron	Tennessee
3	Fred	New York
4	Tom	LA
5	Mary	Georgia

Foreign_key

Order ID	Customer_ID	Order Date	Shipping Status
01	1	1/1/2022	Delivered
02	1	9/1/2022	In Progress
03	2	12/20/2022	Not Started
04	3	8/15/2022	Delivered
05	4	5/31/2022	Delivered

"Cascading Actions in SQL"

Cascading Actions

- **Cascading actions in SQL refer to the automatic propagation of changes in a parent table to related child tables through foreign key constraints.**
- **It ensures that actions such as deletion or modification in the parent table automatically affect corresponding records in the child tables.**
- **Cascading actions are used to maintain data integrity and simplify data management by reducing manual updates.**

- **DELETE CASCADE**

- **When DELETE CASCADE is applied to a foreign key relationship, it ensures that when a row with a primary key is deleted from the parent table, the corresponding row in the child table is also deleted.**
- **This action helps maintain consistency and prevents orphaned records in the child table.**

- **UPDATE CASCADE**

- **When UPDATE CASCADE is applied to a foreign key relationship, it ensures that when a referencing row with a primary key is updated in the parent table, the same update is applied to the child table as well.**
- **This action ensures that changes made to the parent table are reflected consistently in the child table.**

Benefits of Using Foreign Keys

- **Using foreign keys eliminates the need to store data repeatedly in multiple tables.**
- **Directly referencing primary keys in another table reduces data redundancy and improves data integrity.**
- **Foreign keys establish relationships between tables, enabling efficient data retrieval and maintaining data consistency.**

"Database Relationships and Types"

One-to-one relationships

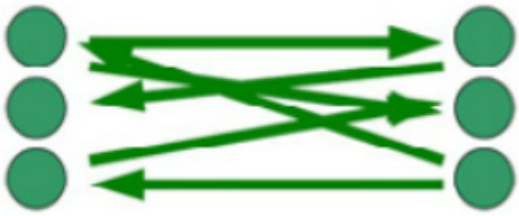
A one-to-one relationship is a type of relationship between two tables in which one record in one table is associated with only one record in another table.

Example
In a school database, each student is assigned a unique student_ID, and each student_ID is linked to only one student.

In a country database, each country is associated with one capital city, and each capital city is associated with only one country.

Many-to-many relationships

A many-to-many relationship is one in which multiple records in one table are associated with multiple records in another table.



Example
Let's consider the *Customers* **and** *Products* **tables. Customers can purchase multiple products, and each product can be purchased by different customers.**

In a relational database, a direct many-to-many relationship is typically not permitted between two tables. For example, in a bank transaction database with multiple invoices having the same number, it can be difficult to map the correct invoice and retrieve the necessary information when a customer makes an inquiry. To address this issue, many-to-many relation tables are often broken down into two one-to-one relationship tables by introducing a third table known as a join table. The join table holds the primary key of both tables as a foreign key and may also contain other necessary attributes.

For example, let's consider the *Products* **and** *Distributor* **tables.**

Here we have the following attributes in each table.

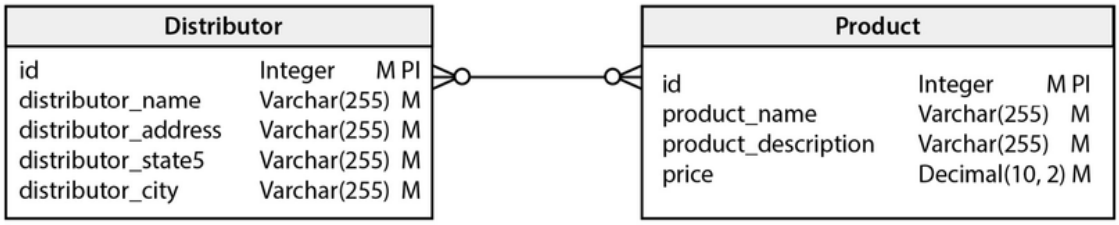
The attributes for the *Distributor* **table are as follows:**

- *id*: **The distributor's ID is the primary key used to identify the distributor**
- *distributors_name*: **The distributor's name**
- *distributors_address*: **The distributor's address**
- *distributors_city*: **The city where the distributor is located**
- *distributors_state*: **The state where the distributor is located**

The attributes for the *Products* table are as follows:

- *id*: **The product's ID is the primary key used to identify the product ID**
- *product_name*: **The product's name**
- *product_description*: **The product's description**
- *price*: **The product's price per unit**

Multiple products can be ordered by multiple distributors, and each distributor can order different products. Therefore, this information needs to be transformed into a relational database model, which would look something like this:



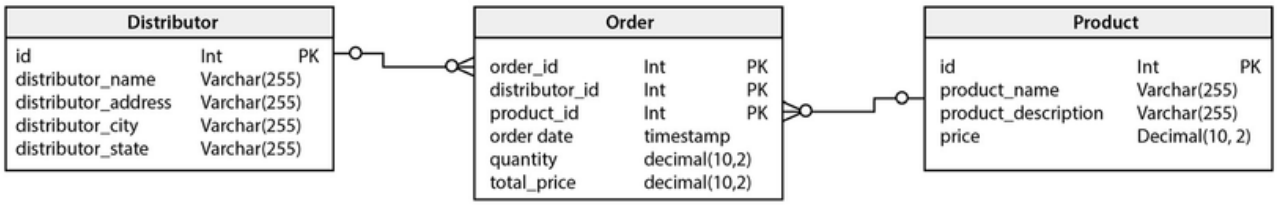
This is known as a join table, and it contains two attributes that serve as foreign keys referencing the primary keys of the original tables.

id **references the distributor ID from the distributor table.**

product_id **references the** *product_id* **column in the** *product* **table.**

These two together serve as the primary key for this table.

However, this information is not sufficient. It would be better to add more attributes to this table.



So, we have now converted it into a relational database model and have the data in a cleaner form. Additionally, I have changed the table name to make it more aligned with the data and named it Orders. We have also added the following additional attributes to the table:

- *order_id*: **Contains the unique order ID for each order placed by the customer**
- *distributor_id:* **The distributor's ID is the unique identifier for each distributor**
- *product_id*: **The unique identifier for each product, which can be ordered**
- *order_date*: **The order date**
- *quantity*: **The number of units ordered**
- *total_price:* **The total price of the orders**

Comparing database normalization and denormalization

Normalization

Database design is a process that aims to reduce data redundancy and maintain data integrity. This systematic approach involves removing data redundancy and undesirable characteristics, such as data insertion, update, and delete anomalies. It achieves this by breaking down larger tables into smaller ones and linking them based on relationships to store data logically.

Data normalization is necessary to eliminate the following anomalies.

- ### Insertion anomalies

An insertion anomaly occurs in relational databases when we are unable to insert data into the database due to missing attributes or data. This is a common scenario where a foreign key cannot be NULL but does not have the necessary data.

For instance, suppose a customer has a customer ID as a foreign key in the Orders table, and no customers have been inserted yet because they haven't ordered any products yet. Therefore, it is impossible to insert a customer who hasn't ordered anything yet.

- **Update anomalies**

An update anomaly occurs when we partially update data in a database. This is a common scenario where data is not normalized, and hence, data elements can reference the same data element in more than one place. As all these different locations are not updated automatically, it is important to manually update this data element at each location.

For instance, suppose we have a Customers table with five columns, including Customer Phone Number and Customer Address. If a customer's address or phone number changes, we must update the table. However, if the table is not normalized, a single customer may have multiple entries, and updating all of them could result in an update anomaly if one of them is overlooked.

- **Delete anomalies**

Data loss can occur when important information is accidentally deleted along with other data. This can result in the loss of crucial data.

For example, let's consider a customer named Adam who ordered one product, a phone. If the order is canceled and you delete the customer from the order table, it will also delete the product (the phone).

Types of normalization

There are four types of normalization:

- **First Normal Form (1NF)**
- **Second Normal Form (2NF)**
- **Third Normal Form (3NF)**
- **Boyce–Codd Normal Form (BCNF)**

1NF

In 1NF, each attribute in the table should contain only atomic values. This means that each cell in the table should hold only one value, and the intersection of rows and columns should not contain multiple values or repeating groups.

For example, in the following table, we have details of the products purchased by customers.

Customer	Customer_Name	Products
1	Adam	Phone, Pen
2	James	Car, Ipod
3	Paul	Laptop, Cup

We can see that the Products attribute holds information related to multiple products purchased by the customer and is therefore not normalized. This is because it contains more than one value in a single cell.

Hence, this can be normalized in the following way in 1NF.

Customer ID	Customer_Name	Products
1	Adam	Phone
1	Adam	Pen
2	James	Car
2	James	Ipod
3	Paul	Laptop
3	Paul	Cup

2NF

There are two rules that must be followed to normalize a table into 2NF:

- **Rule 1 – the table first has to be in 1NF.**
- **Rule 2 – the table should not have any partial dependency.**

This means that every non-prime attribute (all attributes other than the primary key) must be dependent on the primary key of the table.

Whenever the table represents data for two different entities instead of just one entity, it needs to be broken down into its own entity in a different table.

Customer_ID	Customer_Name	Customer_Phone_Number	Order_ID	Order_Status
1	Adam	485-000-9890	1	In Progress
1	Adam	585-000-9890	2	Delivered
2	James	685-000-9890	3	In Progress
2	James	785-000-9890	4	In Progress
3	Paul	885-000-9890	5	Delivered
3	Paul	985-000-9890	6	Delivered

Customer_ID	Customer_Name	Customer_Phone_Number
1	Adam	485-000-9890
1	Adam	585-000-9890
2	James	685-000-9890
2	James	785-000-9890
3	Paul	885-000-9890
3	Paul	985-000-9890

Order_ID	Order_Status
1	In Progress
2	Delivered
3	In Progress
4	In Progress
5	Delivered
6	Delivered

3NF

3NF ensures a reduction in data duplication, thereby maintaining data integrity by following these rules:

- **Rule 1 – the table has to be in 1NF and 2NF.**
- **Rule 2 – the table should not have transitive functional dependencies. This means that non-prime attributes, which are attributes that are not part of the candidate key, should not be dependent on other non-prime attributes within the table.**

In the following table, Customer_ID determines Product_ID, and Product_ID determines Product_Name. Hence, Customer_ID determines Product_Name through Product_ID. This means the table has a transitive dependency and is not in 3NF.

Therefore, the table has been divided into two separate tables to achieve 3NF.

Customer_ID	Customer_Name	Product_ID	Product_Name	Customer_Phone_Number
1	Adam	1	Phone	485-000-9890
1	Adam	2	Pen	585-000-9890
2	James	3	Car	685-000-9890
2	James	4	iPod	785-000-9890
3	Paul	5	Laptop	885-000-9890
3	Paul	6	Cup	985-000-9890

Customer_ID	Customer_Name	Customer_Phone_Number
1	Adam	485-000-9890
1	Adam	585-000-9890
2	James	685-000-9890
2	James	785-000-9890
3	Paul	885-000-9890
3	Paul	985-000-9890

Product_ID	Product_Name
1	Phone
2	Pen
3	Car
4	iPod
5	Laptop
6	Cup

BCNF

BCNF is sometimes referred to as 3.5 NF. Even if a table is in 3NF, there may still be anomalies present if there is more than one candidate key.

Two rules are to be followed for the table to be in BCNF:

- **Rule 1 – the table should be in 1NF, 2NF, and 3NF**
- **Rule 2 – for every functional dependency, such as A -> B, A should be the super key**

Student_ID	Subject	Faculty
Student_1	Cloud Computing	Professor A
Student_2	Big Data	Professor B
Student_3	Statistics	Professor C
Student_4	Project Management	Professor D
Student_5	Analytics	Professor E

There are a few key points to understand here:

- **The preceding table is in 1NF as all rows are atomic**
- **The preceding table is in 2NF as there is no partial dependency**
- **The preceding table is also in 3NF as there is no transitive dependency**

This table does not satisfy the BCNF condition because there is a dependency of Faculty on the Subject, which makes the Faculty column a non-prime attribute while Subject is a prime attribute.

How to fix this

A new super key called Faculty_ID will be introduced, and the preceding table will be split into two different tables, as follows.

Student_ID	Faculty_ID
Student_1	Faculty_ID1
Student_2	Faculty_ID2
Student_3	Faculty_ID3
Student_4	Faculty_ID4
Student_5	Faculty_ID5

Faculty_ID	Faculty	Subject
Faculty_ID1	Professor A	Cloud Computing
Faculty_ID2	Professor B	Big Data
Faculty_ID3	Professor C	Statistics
Faculty_ID4	Professor D	Project Management
Faculty_ID5	Professor E	Analytics

Denormalization

Introduction to Denormalization

- **Denormalization is the opposite of normalization in the context of database design.**
- **While normalization involves organizing data to maintain data integrity, eliminate redundancy, and facilitate future updates, denormalization trades off some organization for faster data retrieval.**

Denormalization in a Nutshell

Denormalization involves storing the same data in multiple locations, leading to redundancy.
The primary aim of denormalization is to improve data retrieval speed, especially when normalized data retrieval is slower due to the need to query multiple tables.

Example

Consider two tables: "Customers" and "Orders," both of which are normalized.
In denormalized tables, data from multiple sources (e.g., customer statistics, IT incidents) are combined into single tables.
The denormalized tables may include redundant customer and order information, but they provide faster data retrieval as no joins are required.

Use Cases for Denormalization

- **Situations where denormalization may be appropriate:**
  a. **Performance improvement: To boost query performance by reading data from fewer tables.**
  b. **Maintaining history: When preserving historical data and validation is required.**
- **Denormalized tables can be useful in monitoring customer service metrics for e-commerce companies.**

Disadvantages of Denormalization

- **Drawbacks of denormalization:**
  a. **Expensive updates and inserts: Data updates in one table must be propagated to other related denormalized tables.**
  b. **Expensive storage: Storing data redundantly across multiple tables consumes more storage space.**
  c. **Data inconsistency: Incorrect data updates may lead to data discrepancies and errors.**

Data Profiling and Preparation before Data Wrangling

- **Data wrangling and its importance**
- **Structured and unstructured data**
- **Data wrangling tools that are used in the industry**
- **What is data profiling?**

Data Wrangling

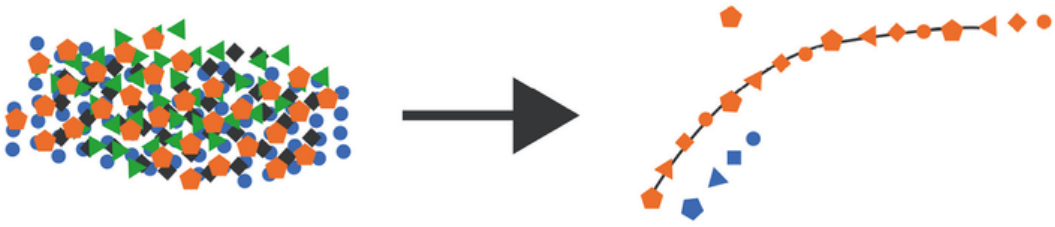
- ***Definition:*** **Data wrangling is the process of cleaning, transforming, and organizing messy data into clean, usable data for generating insights.**
- **It is essential for enabling informed decision-making.**
- **Real-world data is often messy and unstructured, necessitating cleaning before analysis.**

- **Data wrangling includes:**

1. **Cleaning dirty data (missing values, bad characters, unmatched data types, bad formatting).**
2. **Combining datasets from multiple sources.**
3. **Deleting unnecessary data.**
4. **Identifying and handling outliers.**

- **EXAMPLE SCENARIO**

For instance, consider a hotel database (Xostl) with inconsistent customer data. Data wrangling makes it reliable and accurate for analysis.



Data Wrangling Steps

Discovery

- *First Step*
- **Identifying missing values, bad characters, unmatched data types, and data patterns.**

Example Scenario: Understanding customer data by exploring purchases, product history, and data patterns. Key to understanding data from a business perspective.

Structuring

- **In this step, data is transformed from raw format to a better, cleaner structure based on patterns observed in the Discovery step.**

Example: Organizing customer data from various Excel files into a consistent format.

Customer_ID	Name	Address	Phone	Gender	Email
1	Joey	Texas	902-834-2345	M	
2	Ron	Tennessee	0	M	
3	Fred	New York	902-876-5678	M	
4	Kim	LA	902-765-7654	F	
5	Mary	Georgia	0	F	

In the Customers table, we can clearly see that not all the column headings are clear, and they can be structured into a better format

Cleaning

- **Data cleaning aims to remove errors that hinder meaningful inferences.**
- **Tasks include removing outliers, dealing with missing values, standardizing data.**

Cleaning stage example: In the Cleaning stage, we may need to remove outliers, delete data such as missing values, standardize the data using imputation techniques, and various other tasks.

Enriching

Once the data is structured and cleaned into a more usable format, you need to check whether you have all the data required for the project or whether you need to enrich data by extracting values from different datasets. This process is known as the enrichment of data. If you wish to continue with the Enriching step, you need to make sure the previous three steps are followed for the data extracted from different datasets.

Example scenario
Getting additional data, such as the shipping status of various orders of customers from a different database, can lead to better insights.

Validating

Data validation is the process of ensuring data has the right quality and consistency.

During this process, you need to make sure several different checks are done to ensure the highest quality of the data.

Several different checks can be done:

- **Missing values**
- **Duplicates**
- **Bad characters**
- **Data types**
- **Formatting**
- **Logical conditions**
- **Data consistency**

Publishing

- **The final step after following preceding steps, ensuring data is of the highest quality.**
- **Published data is made available to individuals within the organization, based on security access.**
- **The method of sharing data depends on project goals and can include dashboards, reports, Excel files, etc., facilitating business decision-making.**
- **After this step, the data is ready for analysis.**

Customer_ID	Name	Address	Phone	Gender	Email
1	Joey	Texas	902-834-2345	M	
2	Ron	Tennessee		M	
3	Fred	New York	902-876-5678	M	
4	Kim	LA	902-765-7654	F	
5	Mary	Georgia		F	
6	Tim	Texas	902-834-2345	M	
7	Lee	California	902-834-2346	M	
8	July	Washington	902-876-5678	F	
9	Julia	LA	902-765-7654	F	
10	Sam	Georgia	902-723-7827	F	

Figure 2.6– Customers table after data wrangling

The importance of data wrangling

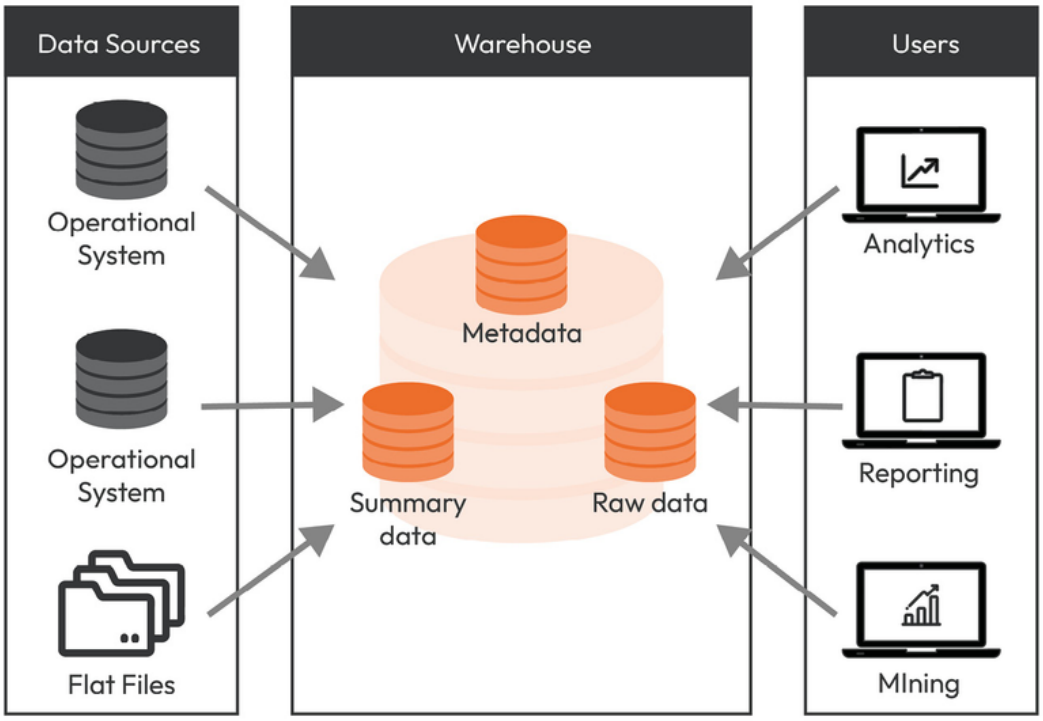
80% of data analysts' time is spent on data wrangling, and just 20% of their time is spent on analysis.

- **Data quality**

Maintaining data quality means ensuring the right data standards are met for analysis and the highest quality is maintained

- **Multiple data sources**

Getting data from multiple different sources into a centralized location requires proper data wrangling to ensure consistency is maintained while combining the data.



Data sources

- **Timesaving**

Without proper systems, this process can be time-consuming. Standardized processes and automation can save a lot of time.

- **Cost effective**

Companies need to build streamlined processes and good practices for data wrangling, which can eventually save lots of money. At the end of the data only if the data is right, better decisions can be made ultimately resulting in good profits for the company.

- **Better decisions**

Using data, stakeholders make important business decisions that can directly impact companies' revenue, profits, and loss. Hence, it's very important to have reliable data to make informed decisions.

- **Growing data**

Data is produced each day and is stored in multiple different structured and unstructured formats. Data wrangling plays a key role in making sense of all these formats, connecting them, and bringing everything into one place.

- **Trust**

With proper systems and processes in place, all the dirty data validations can be systematically handled, hence building consistency and trust in the data. Trust in data is an important aspect of decision-making and generating insights. Data wrangling helps to build that trust.

- **Automation**

More and more companies are adopting streamlined processes and automation pipelines for data wrangling by using different data wrangling tools and moving away from the legacy approach of manually transforming the data. Real-time analytics requires data instantly, and automated data wrangling helps to generate clean and usable data within seconds.

Data wrangling use cases

- **Integrating data from multiple data sources into a single dataset for analysis**
- **Identifying missing values in the data and either imputing or deleting them**
- **Identifying inconsistent formats in the data and cleaning that data**
- **Ensuring consistent data types are maintained while combining datasets**
- **Adding logical checks in the data wrangling process for validation purposes**
- **Detecting outliers in the data and either removing them or taking some action to build consistent data**

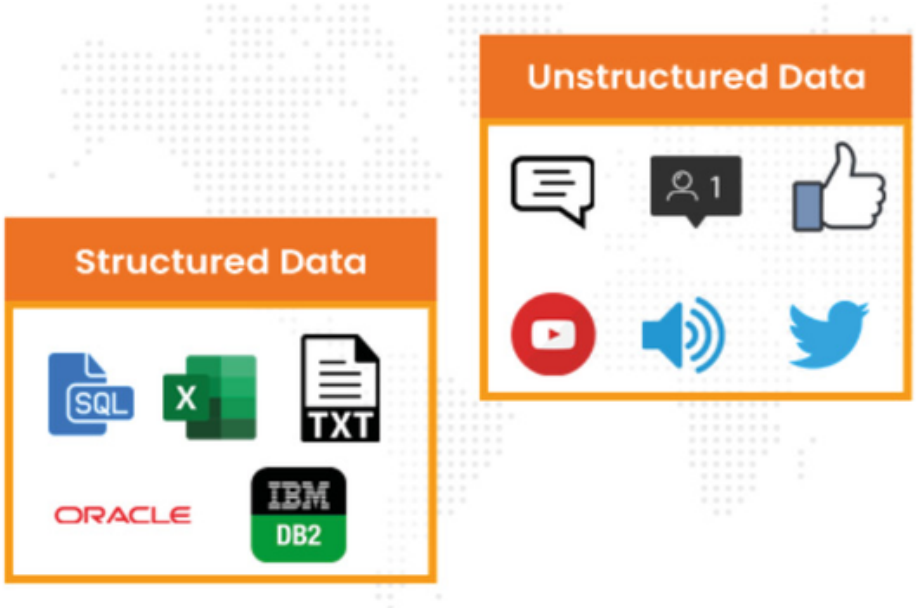
Data-capturing techniques

There are several data-capturing techniques that are used to capture data based on the type of data.

Web scraping

As the name suggests, web scraping is a technique in which we extract data from a website and then save it in a more useful format, such as a CSV, Excel, or JSON file, depending on the type of data we are extracting and the type of data document we are trying to create.

Structured versus unstructured data



Structured	Unstructured
Data stored in rows and columns	Unorganized vast data
Scales slowly	Scales exponentially
Data types – int, Boolean, string	Data types – images, text, emails, messages, feedback reviews
Can be easily used by non-technical users as well	Cannot be used by non-technical users easily as it's complex and needs transformation
Example: Excel, SQL database	Example: Survey results – how many times you use Excel

Data profiling

Data profiling is the process of assessing, analyzing, and building useful summaries of data quality to decide whether the data can be used for any particular project.

These are the main objectives of data profiling:

- **Validating descriptive statistics of the data, such as min, max, count, and sum**
- **Validating different data types and lengths of the data**
- **Validating trends and patterns**
- **Performing data quality checks**
- **Validating different join conditions and how they perform**
- **Finding out metadata and assessing its quality**
- **Identifying different dependencies between tables**

Data profiling techniques

- **Column profiling**

This is the process of extracting the total number of times a value occurs in each column. This can be an important way to identify the most frequent values and patterns within the data.

- **Cross-column profiling**

This technique is used to look at the different columns to validate dependencies and keys between different columns. This step is useful to identify any valid primary keys and dependencies within the dataset, which can then be used to identify dependencies between different datasets.

- **Cross-table processing**

This technique is used to look at different tables to identify foreign key relationships. This can be used to identify relationships between tables and can help understand how to map the data correctly.