


Understanding Relational Database — Part 1

Understanding concepts of database Relational Model.

 Kueila Ramos · Follow
9 min read · Apr 2, 2021

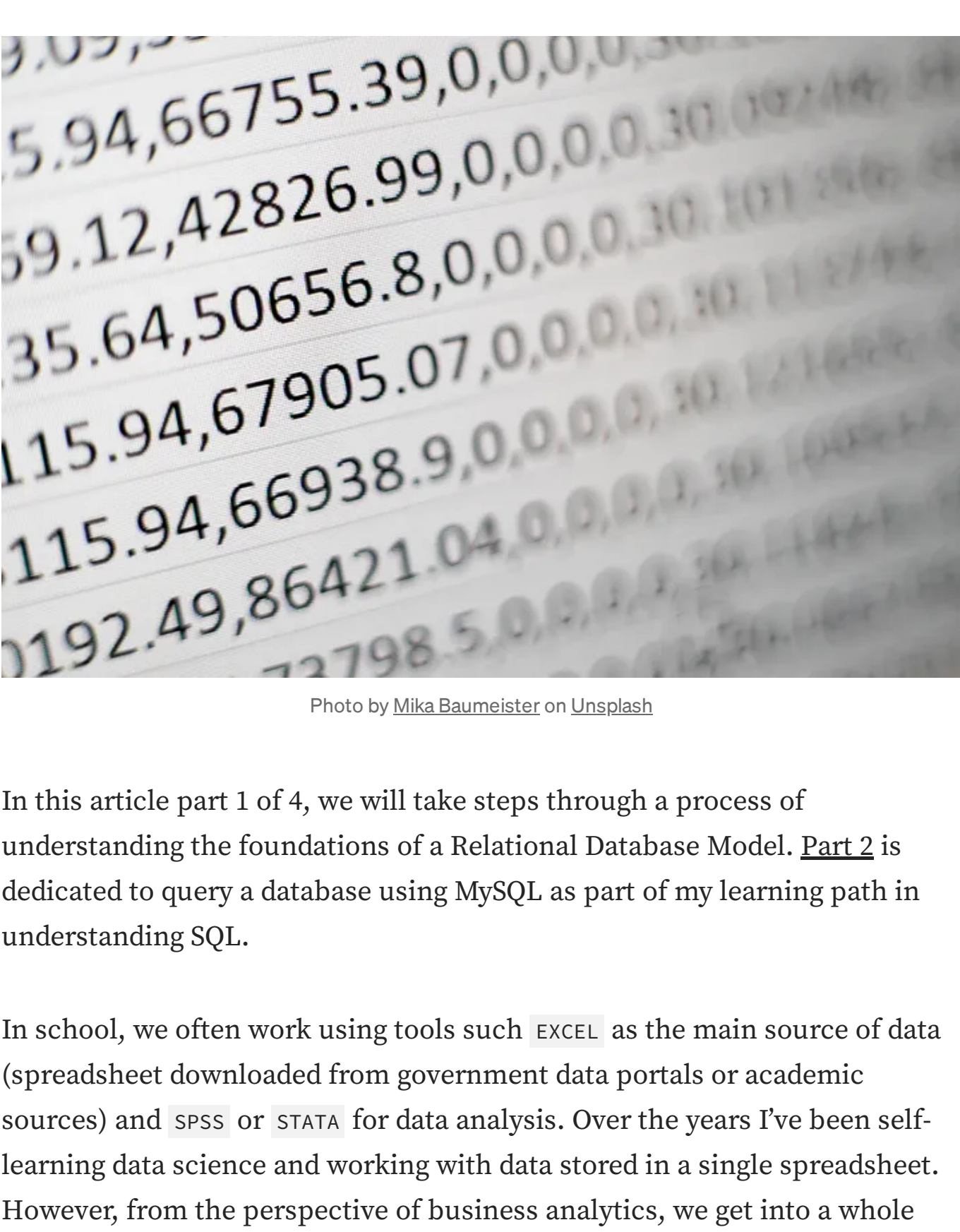


Photo by Mita Baurostator on Unsplash

In this article part 1 of 4, we will take steps through a process of understanding the foundations of a Relational Database Model. [Part 2](#) is dedicated to query a database using MySQL as part of my learning path in understanding SQL.

In school, we often work using tools such as `EXCEL` as the main source of data (spreadsheet downloaded from government data portals or academic sources) and `SPSS` or `STATA` for data analysis. Over the years I've been self-learning data science and working with data stored in a single spreadsheet. However, from the perspective of business analytics, we get into a whole different world. The overwhelming amount of data we need reside in databases.

When we talk about databases is a completely different picture as it involves a large amount of data and at this point, SQL becomes a necessary tool and I found myself struggling to understand some terminologies associate with them as I came from a business/economic background.

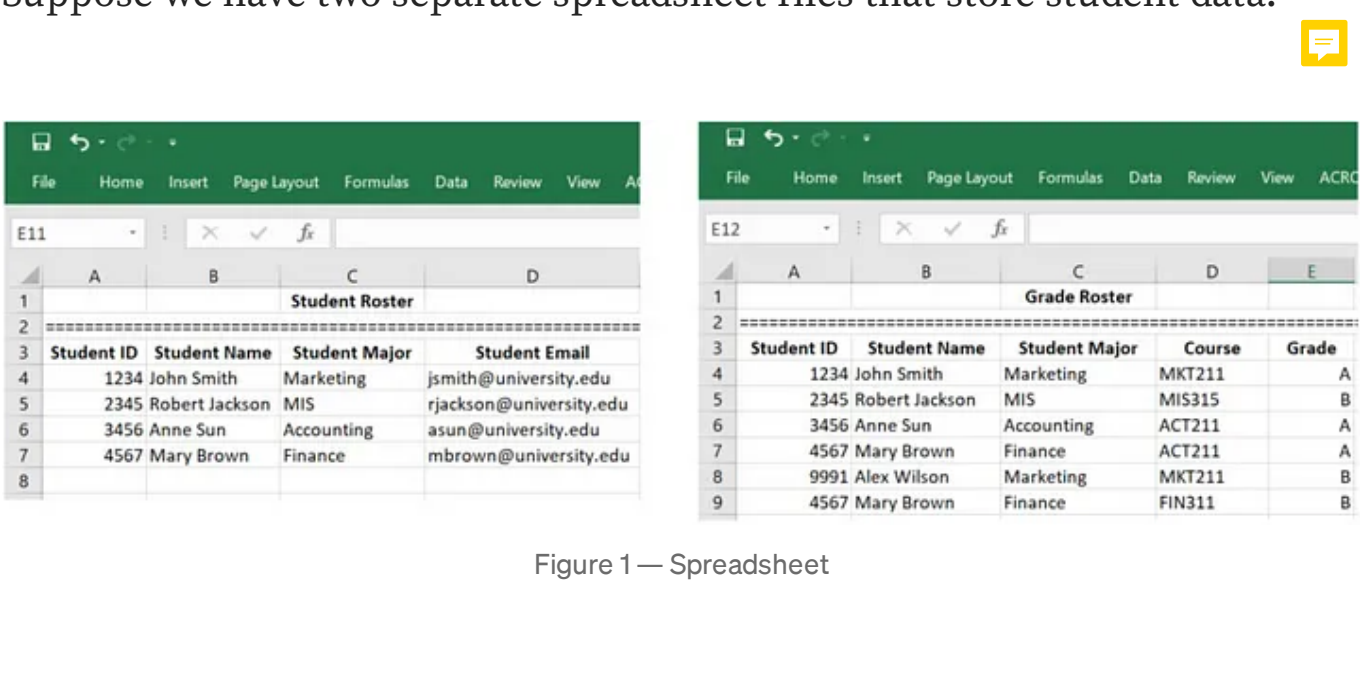
Database's ecosystem includes concepts such as the *Relational Model*, *Data Normalization*, *Entity Relationship Diagram* and so on. As part of my learning process, in this blog, I will firstly look at some elementary database concepts without getting into too much detail, but sufficiently important to understand what is a database and its relationship with SQL. In another blog, I'll use a database for the purpose of understanding these concepts in practice.

Understanding the basic principles of the database in the Relational Model of data

The digital era has brought three related implications: the rapid growth of electronic device, the massive growth of data and rapidly evolving methods for analysing data.

Data has become essential to every business and in almost every organization data have been gathered from several sources. For example, from an e-commerce website perspective, data can be split into two major components: a front-end that displays and collects information, and a back-end for storing the information.

Many people do not know much about database technology (including myself). We usually use an `EXCEL` spreadsheet which often leads to data inconsistency. The use of a spreadsheet may not cause a serious problem for a small data set, however, for a large organization, it could lead to serious errors.



Database vs Spreadsheet

The common problems in spreadsheets are as follow:

- (1) **No control of redundant data:** Some people/organization often keep redundant data for convenience. This redundancy of data could make the data set inconsistent.
- (2) **Violation of data integrity:** Data integrity means consistency among the stored data.

Suppose we have two separate spreadsheet files that store student data:

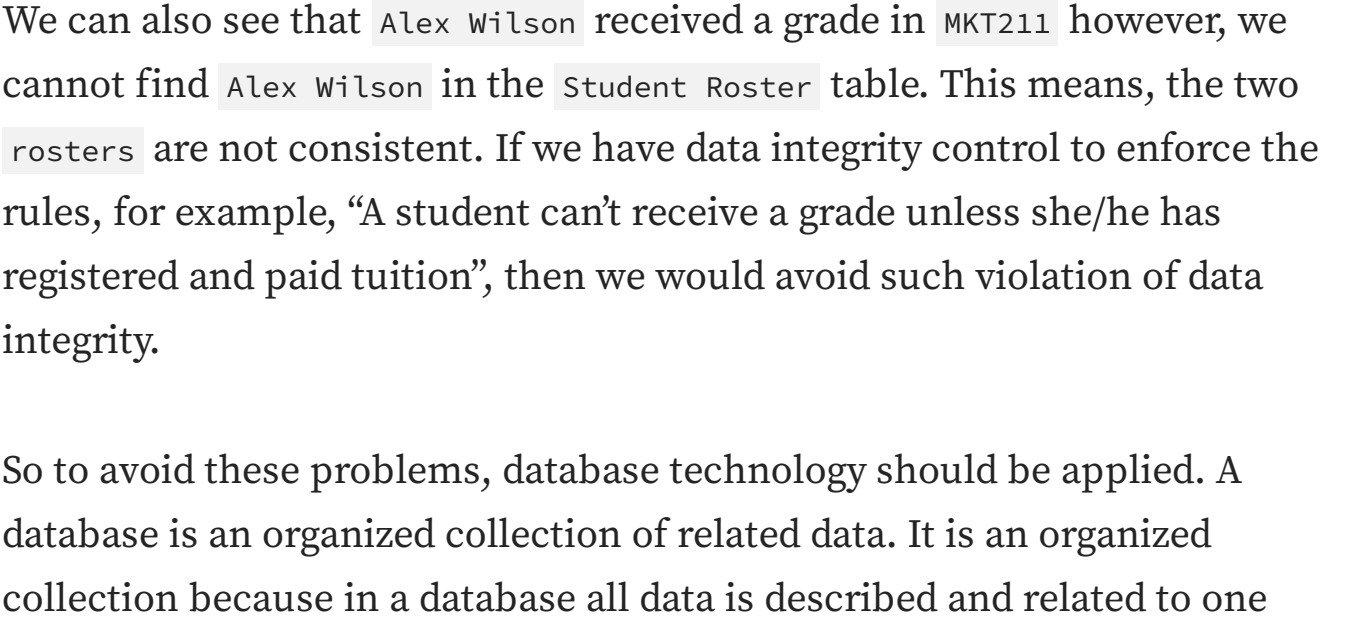


Figure 1 — Spreadsheet

Take a look at the `Grade Roster` table in fig. 1: Student `Mary Brown` with student ID number `4567` is stored more than once. This is called data redundancy, for some people, it could make data access convenient, but it can be harmful. For example, if `Mary Brown` changes her name or her major, then all her names and major stored in the system must be changed altogether.

For small data systems, it could be easier to solve. However, when the data system is huge, making changes to all redundant data is difficult if not impossible. To try to solve the redundancy problem, as a result, the entire data set could be corrupted.

We can also see that `Alex Wilson` received a grade in `MATH1` however, we cannot find `Alex Wilson` in the `Student Roster` table. This means, the two `rosters` are not consistent. If we have data integrity control to enforce the rules, for example, "A student can't receive a grade unless she/he has registered and paid tuition", then we would avoid such violation of data integrity.

So to avoid these problems, database technology should be applied. A database is an organized collection of related data. It is an organized collection because in a database all data is described and related to one another.

There have been different types of databases over time: Hierarchical Database, Network Database, Relational Database, NoSQL Databases and so on. Databases follow a design model and the most popular is called **Relational Model** that was introduced by Edgar F. Codd.

The structure of Relational Model related to databases starts with a concept called tables referring to them as a collection of relations.

In a Relational Model, data are stored in a **two-dimensional table** or **multiple tables related to each other** through special key fields called **Primary Key** and **Foreign Key**. Informally, each relation resembles a table of columns and rows. In a formal relational model terminology, a column is called an **attribute**, a row is called a **tuple** and the table is called a **relation**.

We said before that, one of the key differences between databases and spreadsheets is that spreadsheets do not ensure the integrity of their stored data while databases must enforce data integrity. The following categories are the types of data integrity that must be enforced:

- **Domain integrity:** enforces valid entries for a given attribute (column) by restricting the type, format, or range of values.
- **Entity integrity:** every tuple (rows) has a unique primary key that cannot be null. It simply means that there are no duplicate rows.
- **Referential integrity:** means that every Foreign Key in a secondary table matches a primary key that is in the parent table.

You may be asking, how databases can ensure data integrity?

One important concept to understand is **Normalization**. In simple terms, to normalize a database means to design it in a way that:

- 1) Removes data redundancy
- 2) Ensure data integrity

Let's take a practical example:

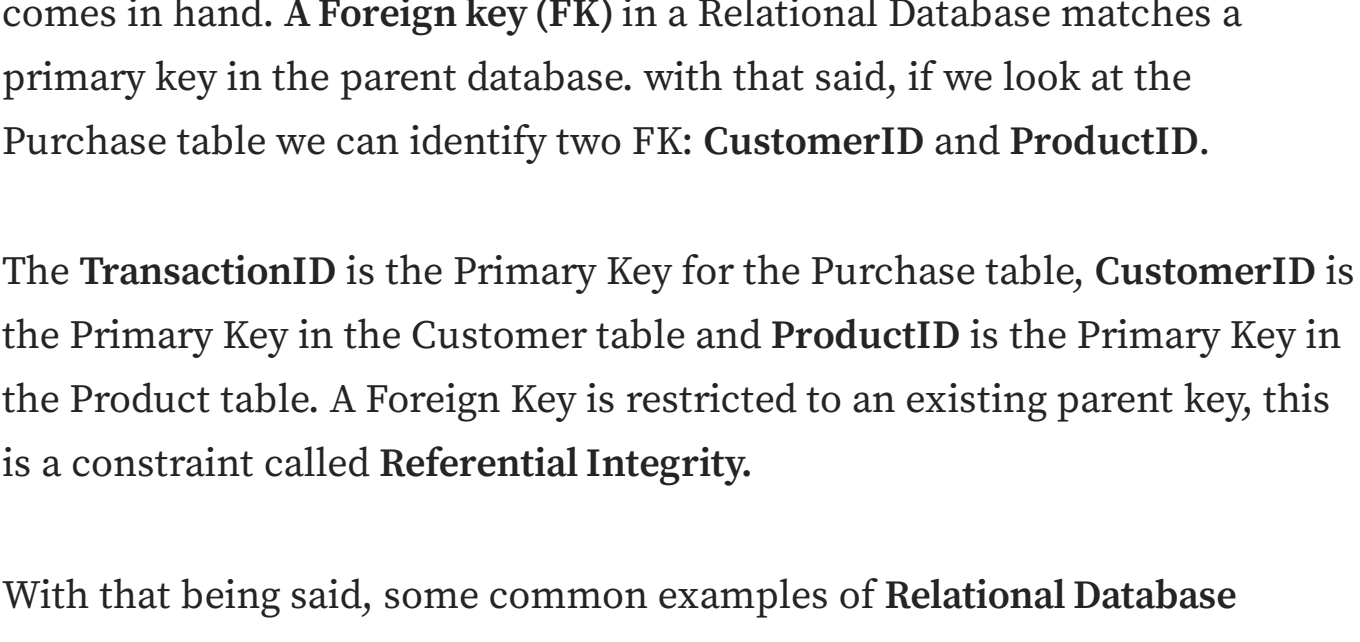


Figure 2 — Google source with some adaptation

Fig. 2 shows a Relational Database for three tables: Customer, Product and Purchase table.

Each table has two properties: tuples (rows) and attributes (columns). The intersection of a tuple and an attribute in a database is called a **value**.

An **Attribute**: represents each column in the table and they are the properties that define a relation. For example, from the relation schema above:

- **Customer table** is the entity name of the relation, which have three attributes : `CustomerID`, `Customer Name` and `Address`.
- **Product table** is the name of the relation, which have three attributes: `ProductID`, `Name` and `Price`
- **Purchase table** is the name of the relation, which have four attributes: `TransactionID`, `CustomerID`, `ProductID`, and `Purchase date`.

Degree of relation: is the total number of attributes existing in the relation.

Relation Schema: represents the name of the relation with its attributes.

Cardinality: is defined as the total number of values from the underlying domain present in the table. For example, from the relation schema above all tables has 3 tuples in each one.

Attribute domain: is the original sets of atomic values used to model data. This means that values in tables must have a consistent data type. The relational table itself follows particularly a **defined datatype** such as for example string, date-time, a number that describes the types of values that can appear in each attribute (these are represented by a domain of possible value).

From the perspective of analyzing data integrity:

Looking at figure 2 above, the **Customer table** and **Product table** has a unique primary key identified by ID. Database Management Systems will ensure the integrity of the data stored in those tables. For example, we cannot insert in the customer table the following `24221` because the `CustomerID` value `24221` already exists in the customer table. This constraint is called **Entity Integrity**.

Information about different entities is stored in different tables. To access the data we need a way to navigate between them. Here Foreign Key (FK) comes in hand. A **Foreign key (FK)** in a Relational Database matches a primary key in the parent database. with that said, if we look at the Purchase table we can identify two FK: `CustomerID` and `ProductID`.

The `TransactionID` is the Primary Key for the Purchase table, `CustomerID` is the Primary Key in the Customer table and `ProductID` is the Primary Key in the Product table. A Foreign Key is restricted to an existing parent key, this is a constraint called **Referential Integrity**.

With that being said, some common examples of **Relational Database Management System (RDBMSs)** are Oracle, MS SQL Server, IBM, MySQL, SQLite, Microsoft Access, PostgreSQL, MariaDB to mention a few examples.

Without getting into detail on the nuances that differentiate each Relational Database Management System (which is not the goal of the article), we can say that even though the concepts of SQL is similar, the implementation of Syntax or Keywords is quite different from one to another.

However, they have in common the ability to control the access to a database and enforces database security which relies on confidentiality and integrity of the data stored.

As the database is managed by RDBMS, databases are accessed by a query language called **SQL**.

SQL is a language used to connect or access the data stored in a Relational Database Management System by using syntax that allows us to query relational database by creating objects like, database, tables, read/write and access those tables.

For example, if we work at a company that have databases storing customer information, employees information, sales information etc., as a data analyst user, we could form an SQL query (a question) that asks for exactly what we want to know about those data. The figure below shows us a representative example:

A Database Management System (DBMS) is a computerized system that enables users to create and maintain a database, facilitates the process of defining, constructing, manipulating and sharing databases among various users and applications.

The standard SQL commands to interact with Relational Databases are `CREATE`, `DROP`, `SELECT`, `INSERT`, `UPDATE`, and `DELETE`. These commands can be grouped into 2 subsets:

The **Data Definition Language (DDL)** which are statements used to define the structure of the database, create tables, specify their keys, data types, constraints of the data to be stored in the database and so on. The key, entity integrity, and referential integrity constraints are specified by the Data Definition Language.

The **Data Manipulation Language (DML)** includes functions such as **querying (read)** the database to extract specific data, **updating** the database, **inserting new data** and **deletes data** from a database so, collectively, these four functions are affectionately abbreviated as **CRUD**. The word, *created*, is generally used instead of *insert*.

Conclusion

From a perspective of Relational Database what we can conclude is:

1. Every table must have at least one primary key which that cannot be Null to allows entity integrity
2. Data model must be designed in a way it follows database normalization forms (you may heard about 1NF, 2NF, 3NF forms) to allow us to reduce redundant data in a given database.
3. Referential integrity must be maintained with help of Foreign Key
4. Table design should be organize in a way it follows domain integrity also called semantic integrity. Means that values in tables must have consistent data types.
5. Identify if a database holds an one-to-one relationship, an one-to-many relationship or a many-to-many relationship.
6. Integrity constraints specified on the relational database schema must never be violated.

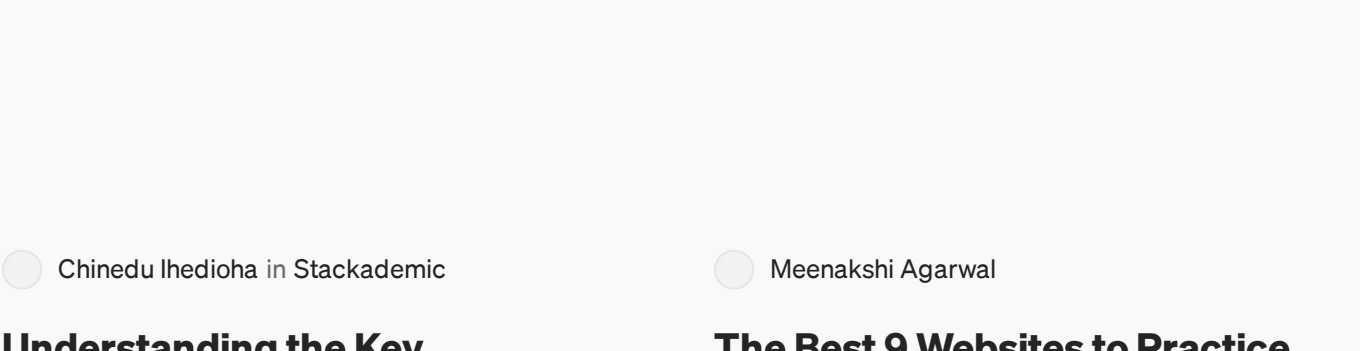
Summary

In this article, we covered some theoretical elements of Relational Database. We explain the difference between database and spreadsheet and important concepts of primary key and foreign key.

The next article will describe the basic SQL query in practice. Thanks for reading, I hope that you liked it and if you have any suggestions please contact me. Catch you on the next blog.

Reference

<https://digitalcommons.biol.edu/open-textbooks/1/>



60 1 15 3

Written by Kueila Ramos · Follow
94 Followers
Tracking my own path to become a Data Scientist in the business field

More from Kueila Ramos

SQL query on a Bank Database —Part 2
This blog is related to part 2 of 3 (see part 1 — Understanding Relational Database). We will...
11 min read · Apr 18, 2021

Web Traffic Analysis with SQL
Analysing web traffic from e-commerce
9 min read · May 10, 2021

Data Preparation & EDA of an E-commerce company using Python
In this article, we are going to analyze an E-commerce dataset that contains transaction...
10 min read · Mar 27, 2021

SQL query with Inner Join, Business questions and visualization with...
Understanding Business requirements by using SQL Inner Join, Venn Diagram and...
7 min read · Apr 19, 2021

See all from Kueila Ramos

Get to Know Closer to Relational Database Management System...
Previously in A Glimpse of DBMS, we learned a little about the Database Management...
6 min read · Oct 21, 2023

Combining Data from Multiple Tables in SQL Without Using JOIN...
3 min read · 4 days ago

Data Warehouse #03
Data Warehouse Architecture
3 min read · Dec 5, 2023

Designing a Banking Database Using SQL and Python
Introduction
3 min read · Oct 27, 2023

Understanding the Key...
The Best 5 Websites to Practice