

System Design in Practice

No SQL Databases



By / Ahmed Khalifa

Index

1. Technology agnostic

- Main reason to choose wrong database
- How to choose the best database for your system

2. SQL Databases

- What is SQL Database ?
- Is Excel sheets program can be a SQL Database ? Why ?
- SQL Database properties :

1. Relations

- The importance of relations for write operations
- The importance of relations for read operations

2. Normalization

- * What is a normalization ?
- * The importance of normalization for read operations
- * The importance of normalization for write operations
- * Is normalization responsibility for database or developers ?
- * Is denormalization always bad ?

3. ACID Principles

- What is ACID ?

- Atomicity :

 - * What is atomicity ?

 - * Atomicity two cases

 - * The importance of atomicity

 - (Transfer money between two bank accounts example)

- Consistency :

 - * What is consistency ?

 - * The importance of consistency

 - (negative balance of bank account example after transfer money example)

- Isolation :

 - * What is isolation ?

 - * The importance of isolation

 - (3 clients try to buy the same product from the same stock at the same time example)

- Durability :

 - * What is Durability ?

 - * The importance of Durability.

3. What are NO SQL Databases ?

4. SQL VS NO SQL Databases

4.1 Data model :

4.1.1 Structure data model

4.1.2 Semi structure data model

4.1.3 Unstructured data model

4.2 Database Language

4.2.1 Standard language

4.2.2 Complex query

4.3 Scalability

4.3.1 Vertical scalability

4.3.2 Horizontal scalability

4.4 Consistency and concurrency model

4.4.1 Strong consistency

4.4.2 Causal consistency

4.4.3 Eventual consistency

4.5 Theorem and principles

4.5.1 ACID

4.5.2 CAP

Technology agnostic :

Main reason to choose wrong database :

One of the major mistakes I see in the software industry is choosing the database based on the technology stack for example:

- If the developers team use **.Net stack** , the database will be automatically **SQL Server**.
- If the developers team use **MEAN stack** , the database will be automatically **MongoDB**.

How to choose the best database for your system ?

1. Different businesses need different databases , banking system needs database that different than social media app needs , you must choose the database that best suitable for your business domain.
2. Non-functional requirements is a very another important factor to choose your database , system of **1000** users needs database that different than system with **10,000,000** users. we will discuss all non-functional requirements on details later.

SQL Databases

SQL databases also known as Relational Databases.

What is SQL Database ?

Database that store data on a tabular format (rows and columns)

Clients Table

Id	Name	IsActive

Orders Table

Id	OrderAddress	ClientId	CreatedDate

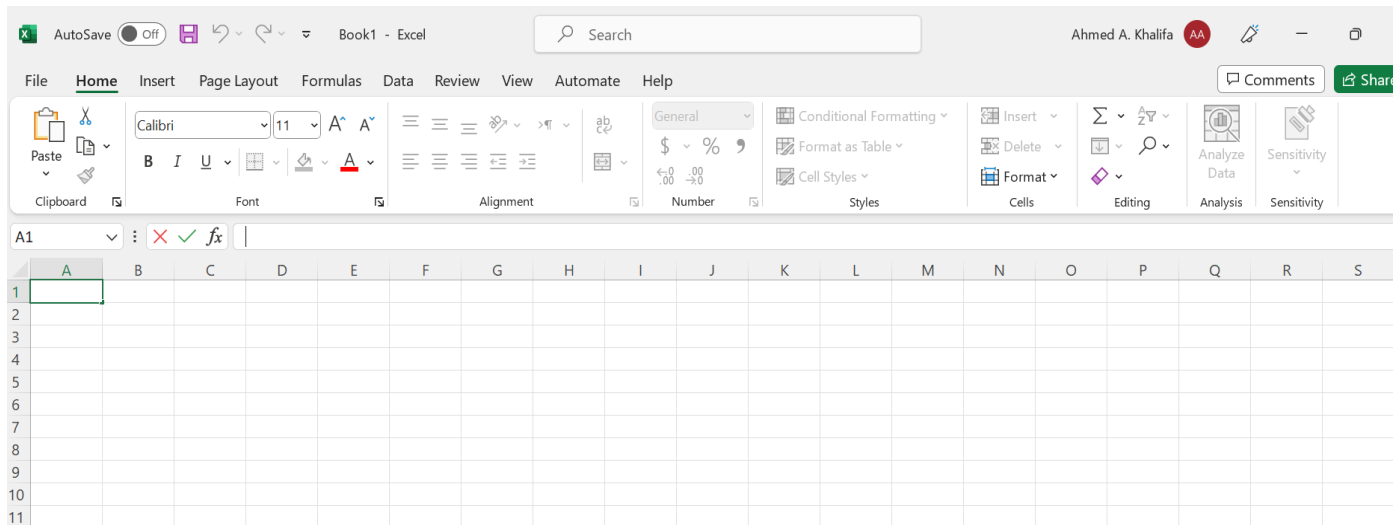
OrderItems Table

Id	OrderId	ProductId	Quantity	UnitPrice

Products Table

Id	Name	Description	CurrentUnitPrice

Is Excel sheets can be SQL Database because it store data on tables (rows – columns) ?



Excel sheets is just a program to store data , **Not** any program or data store use tables, rows and columns will be SQL Database.

SQL Database has a set of standard conditions or properties that must be implemented and here list of the most important (not all) of them :

1- Relations.

2- Normalization.

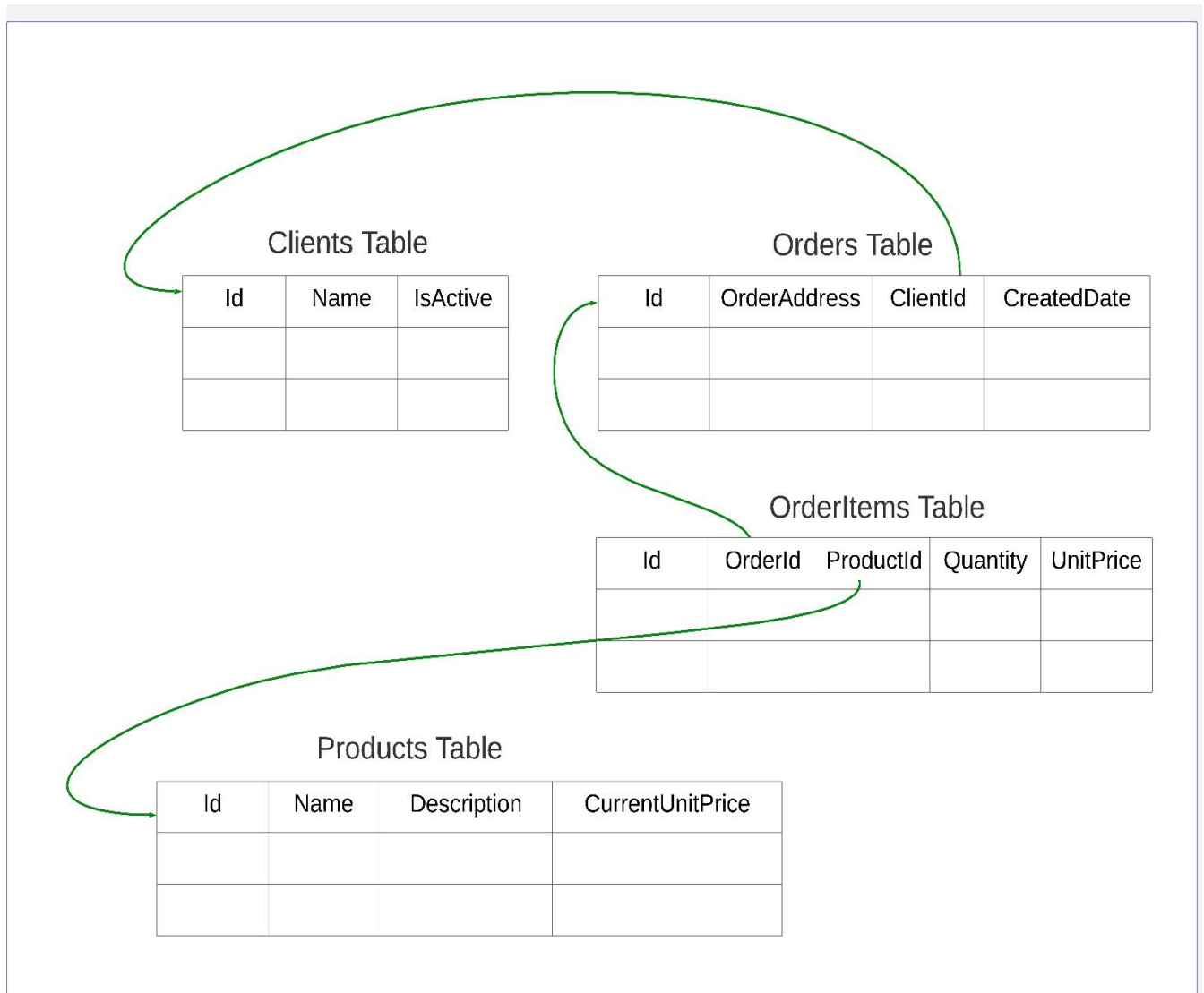
3- ACID.

We will discuss them in more details in next section.

SQL Database Properties :

1.Relations :

The tables have relations between each other using foreign keys.



The importance of relations for write operations :

Relations achieve **data integrity** for write operations (insert – update – delete) :

example 1 : you cannot delete client from banking system who took a loan and did not repay it, or performed some financial transactions .

example 2 : you cannot delete client has orders and there reference to this client id in order table.

DELETE FROM Clients
WHERE Id = 5468



Clients Table

Id	Name	IsActive
5468	Ahmed Khalifa	

Orders Table

Id	OrderAddress	ClientId	CreatedDate
8906	Egypt , Alex	5468	30-08-2023

OrderItems Table

Id	OrderId	ProductId	Quantity	UnitPrice
109876	8906	109	1	1000

Products Table

Id	Name	Description	CurrentUnitPrice
109	iPhone 14	iPhone 14 pro max , 5G , 256 Ram	1000

The importance of relations for read operations :

Join query is one of the main properties on SQL databases.

using Join query : SQL Database can read and display data from different tables in single query.

```
SELECT
Clients.Name as ClientName ,
Orders.OrderAddress as OrderAddress ,
Products.Name as ProductName ,
OrderItems.Quantity as ProductQuantity ,
OrderItems.Quantity as ProductUnitPrice ,
OrderItems.Quantity as ProductTotalPrice ,
FROM Orders
INNER JOIN Clients    ON Clients.Id = Orders.ClientId
INNER JOIN OrderItems ON OrderItems.OrderId = Orders.Id
INNER JOIN Products  ON Products.Id = OrderItems.ProductId
WHERE Orders.Id = 8906
```

ClientName	OrderAddress	ProductName	ProductQuantity	ProductUnitPrice	ProductTotalPrice
Ahmed Khalifa	Egypt , Alex	iPhone 14	2	1000	2000

Clients Table

Id	Name	IsActive
5468	Ahmed Khalifa	

Orders Table

Id	OrderAddress	ClientId	CreatedDate
8906	Egypt , Alex	5468	30-08-2023
....			
8930	Egypt , Cairo	5468	01-09-2023

OrderItems Table

Id	OrderId	ProductId	Quantity	UnitPrice	TotalPrice
109876	8906	109	2	1000	2000
.....					
109999	8930	120	3	1500	4500

Products Table

Id	Name	Description	CurrentUnitPrice
109	iPhone 14	iPhone 14 pro max , 5G , 256 Ram	1000
.....			
120	lenovo laptop	Lenovo Yoga C700 -Intel Core i7-1255 -16GB DDR4-1TB	1500

2.Normalization :

What is a normalization ?

Normalization in SQL Databases is a techniques that aims to prevent data duplications and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies.

Normalization has a lot of forms , but we will discuss only the general concept.

Normalization as general concept is try to divide the database to small tables as possible and then this table will be logically communicate with each other through the relations or foreign keys.

In the previous section (Relations) we see our database consist of a set of different tables that have relations between each other , if we need data from more than one table , we join this tables.

The importance of normalization for read operations :

Different queries need different data.

Example Query 1 : Get Order Details Of Id 8906

```
SELECT
Clients.Name as ClientName ,
Orders.OrderAddress as OrderAddress ,
Products.Name as ProductName ,
OrderItems.Quantity as ProductQuantity ,
OrderItems.Quantity as ProductUnitPrice ,
OrderItems.Quantity as ProductTotalPrice ,
FROM Orders
INNER JOIN Clients    ON Clients.Id = Orders.ClientId
INNER JOIN OrderItems ON OrderItems.OrderId = Orders.Id
INNER JOIN Products  ON Products.Id = OrderItems.ProductId
WHERE Orders.Id = 8906
```

ClientName	OrderAddress	ProductName	ProductQuantity	ProductUnitPrice	ProductTotalPrice
Ahmed Khalifa	Egypt , Alex	iPhone 14	2	1000	2000

Example Query 2 : Get All Orders Of Client 5468

```
SELECT
Clients.Name as ClientName ,
Orders.Id as OrderId ,
Orders.CreatedDate as OrderCreatedDate ,
FROM Orders
INNER JOIN Clients  ON Clients.Id = Orders.ClientId
WHERE Clients.Id = 5468
```

ClientName	OrderId	OrderCreatedDate
Ahmed Khalifa	8906	30-08-2023
Ahmed Khalifa	8930	01-09-2023

The importance of normalization for write operations :

Prevent data duplication using normalization that can cause a lot of conflicts , performance issues , extra storage space for unnecessary data.

For example :

imagine our e-commerce database consist of only single table called orders table contain every thing.

Our client want to update or change his name , in denormalization database , the update command need to loop on all rows to update our client name.

Denormalization Database

Orders Table

Id	OrderAddress	ClientName	CreatedDate	Product	Quantity	UnitPrice	TotalItemPrice
8906	Egypt , Alex	Ahmed Khalifa	30-08-2023	iPhone 14	2	1000	2000
....							
8930	Egypt , Cairo	Ahmed Khalifa	01-09-2023	lenovo laptop	3	1500	4500

UPDATE Orders
SET ClientName = "Ahmed Ahmed Khalifa"
WHERE ClientName = "Ahmed Khalifa"

But when database apply normalization , we need only to update single row , and any join query with this table or row will reflect the new updated value immediately.

Normalization Database

UPDATE Clients
SET ClientName = "Ahmed Ahmed Khalifa"
WHERE Clients.Id = 5468

Clients Table

Id	Name	IsActive
5468	Ahmed Khalifa	

Orders Table

Id	OrderAddress	ClientId	CreatedDate
8906	Egypt , Alex	5468	30-08-2023
....			
8930	Egypt , Cairo	5468	01-09-2023

OrderItems Table

Id	OrderId	ProductId	Quantity	UnitPrice	TotalPrice
109876	8906	109	2	1000	2000
.....					
109999	8930	120	3	1500	4500

Products Table

Id	Name	Description	CurrentUnitPrice
109	iPhone 14	iPhone 14 pro max , 5G , 256 Ram	1000
.....			
120	lenovo laptop	Lenovo Yoga C700 -Intel Core i7-1255 -16GB DDR4-1TB	1500

Is normalization responsibility for database or developers ?

Most of SQL Databases properties like ACID , Join query , data integrity are responsibility of database or RDBMS.

But normalization is responsibility of developers and their design.

Is denormalization always bad ?

In some cases denormalization can be good but developers must take care of the effect sides of it.

Denormalization can good for :

some business cases for historical data :

if you note in our database design Products table has column called CurrentUnitPrice and OrderItems Table has column called UnitPrice , this because product price can change on the future , the old orders with this product must keep the old price.

3.ACID Principles :

What is ACID ?

4 Properties that define the behavior of the database transaction.

They are **Atomicity, Consistency, Isolation, Durability.**

A → (Atomicity)

What is Atomicity ?

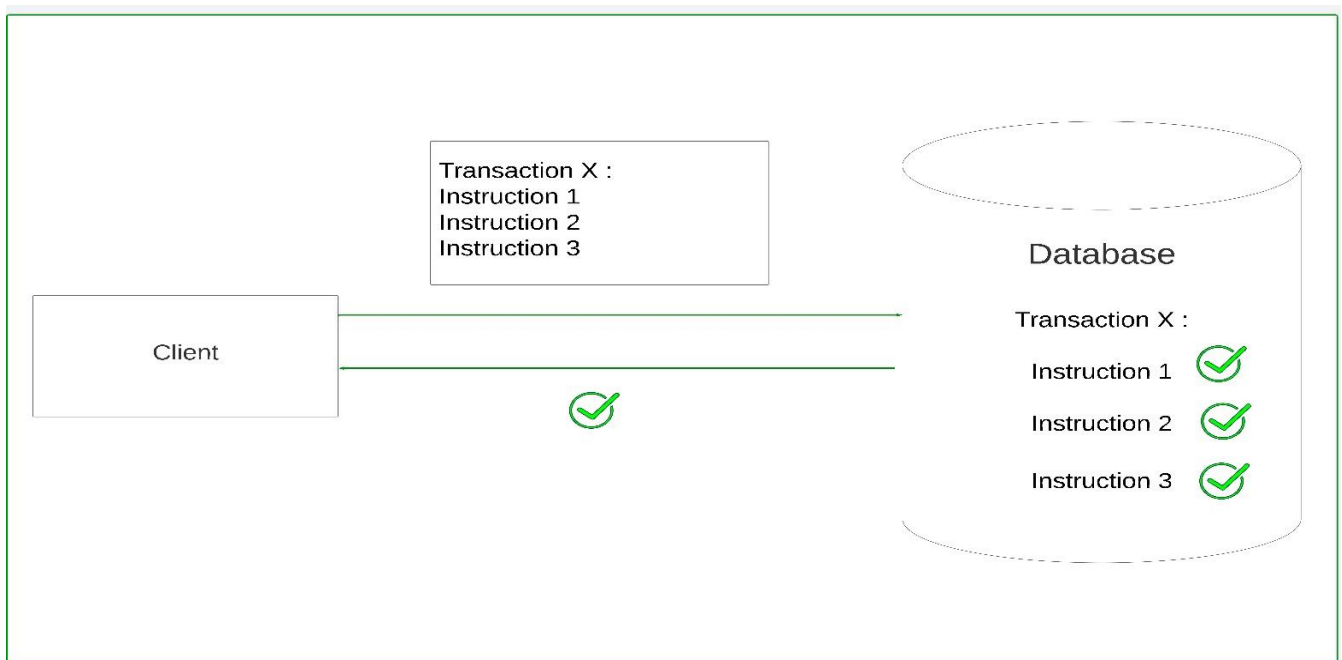
transaction work as a unit of work.

Atomicity two cases :

Transaction has only two cases :

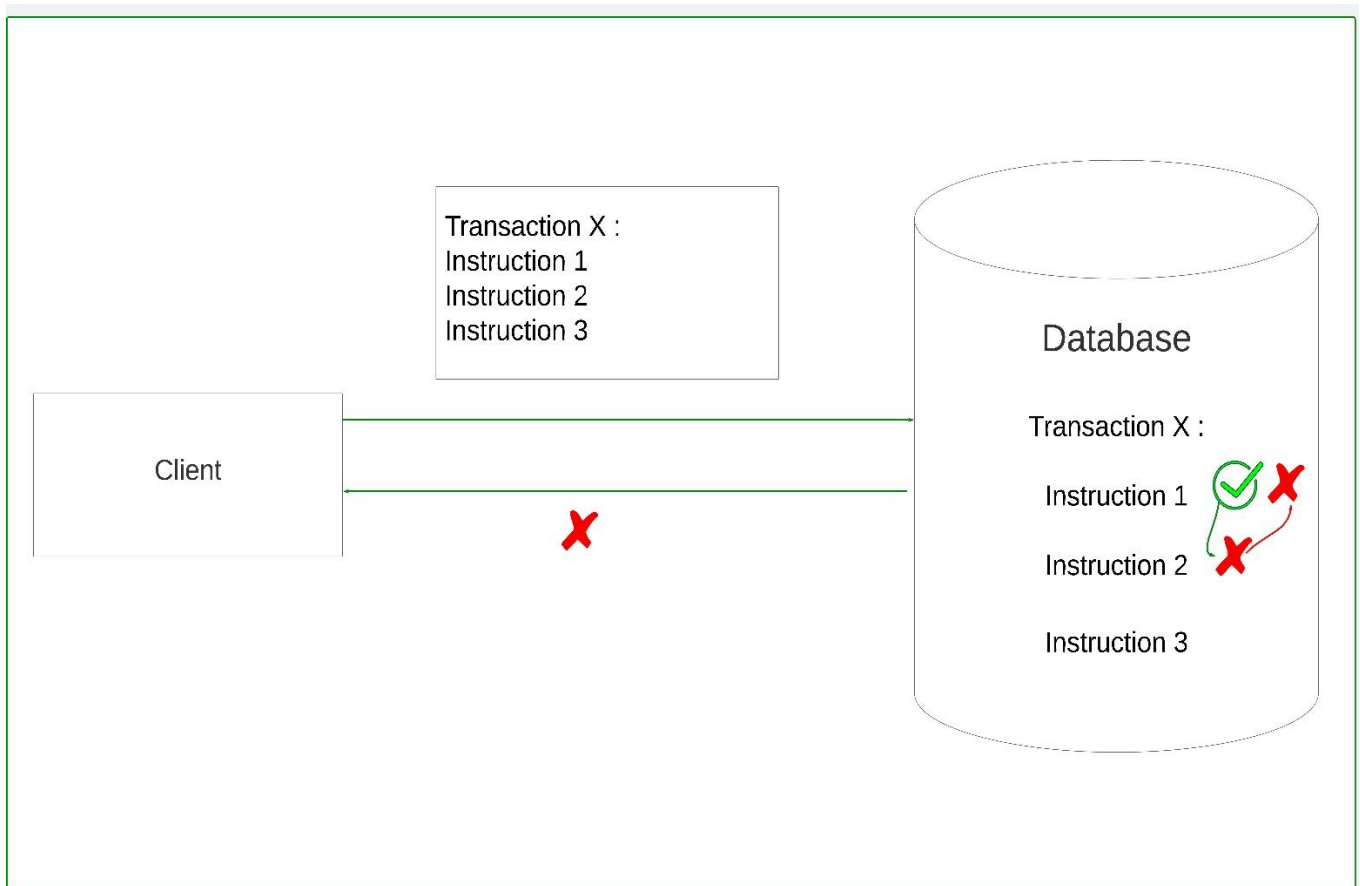
Case 1 if no failure :

All instructions or statements executed successfully.



Case 2 If there are failure at any instruction :

- all instructions or statements not executed.
- database should abort or rollback any instruction executed before this failure.
- then return errors to the client.



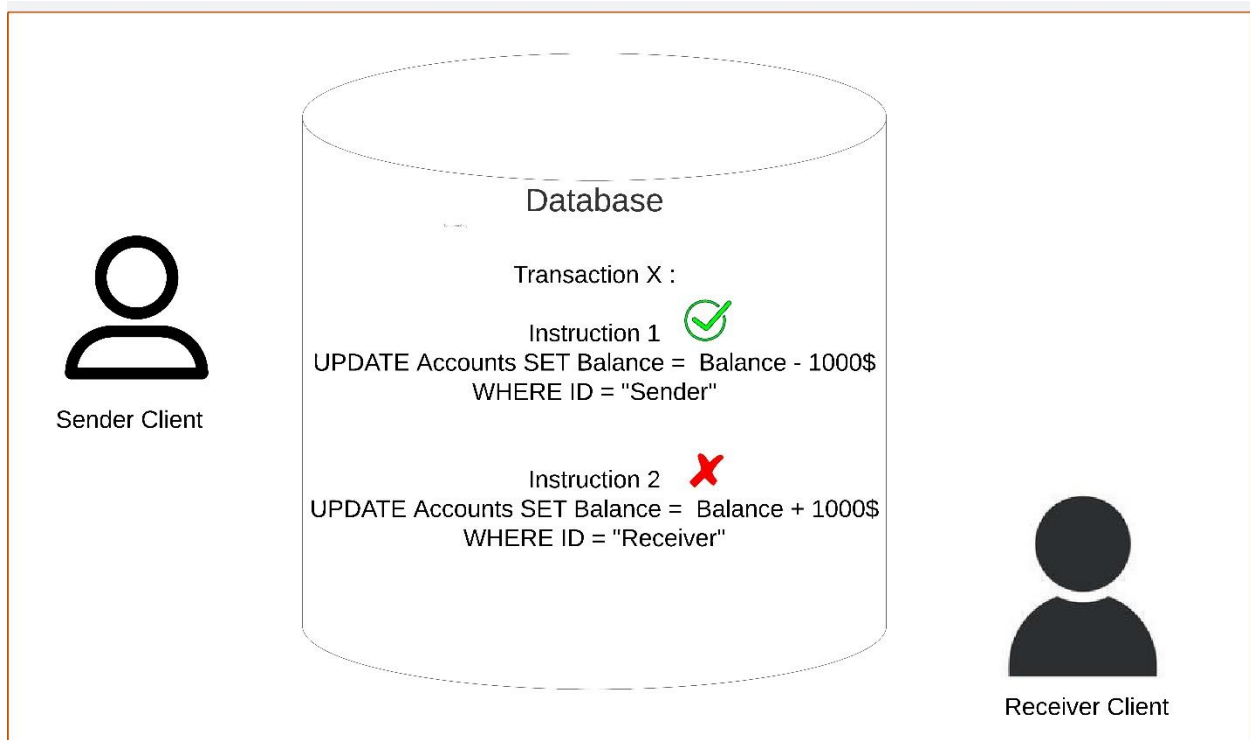
The importance of atomicity

To understand the importance of the atomicity we will discuss a very famous example :

transfer money between two bank accounts

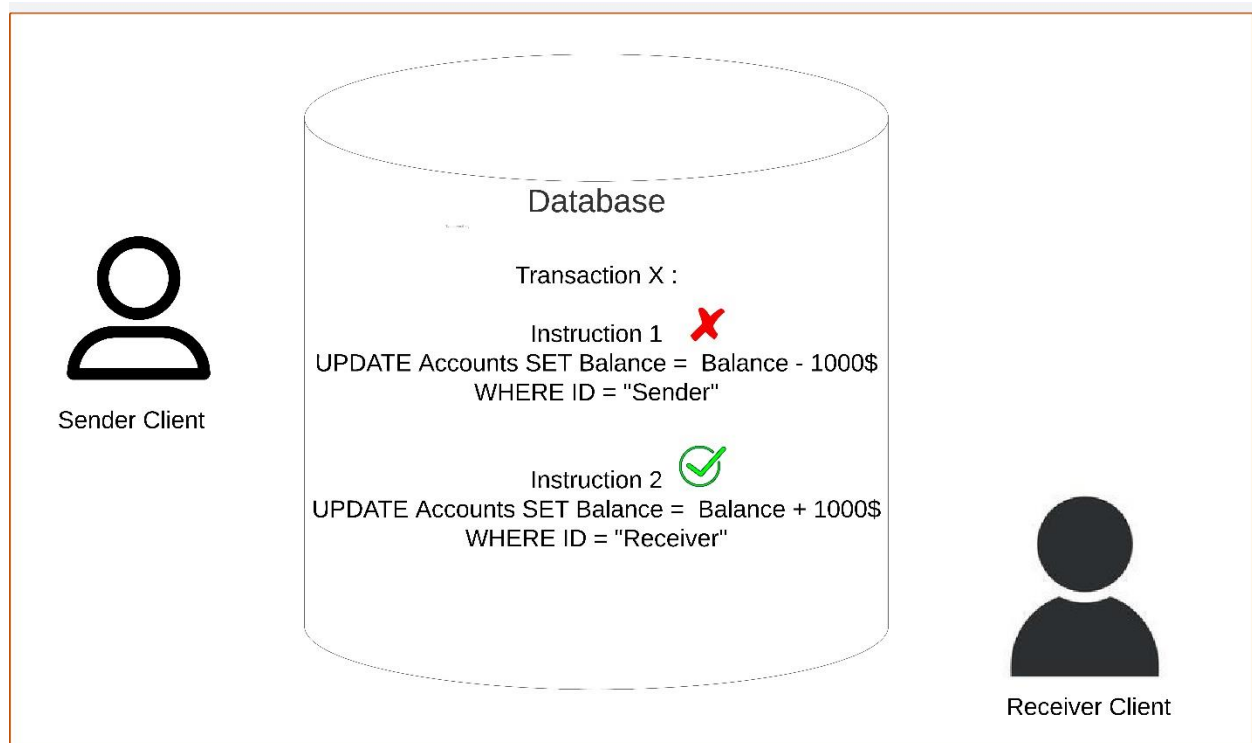
If the database does not support the atomicity :
case 1 :

The transaction reduce the money from the client 1 account , and not sending it to the client 2 account , so client 1 loss his money.



case 2 :

the transaction does not reduce the money from client 1 account , but send it to client 2 account , so the bank will lose its money on this case.



C → (Consistency)

Consistency in ACID Theorem has a different definition than Consistency on CAP Theorem or distributed systems , we will discuss consistency in CAP Theorem later in details.

What is Consistency in ACID ?

Transaction must move the database from valid state to another valid state.

In simplicity consistency mean all write operations or instructions in our transaction must apply all defined rules like : constraints , cascades , triggers , referential integrity and .. etc.

The importance of Consistency with example :

Negative balance bank account after transfer money

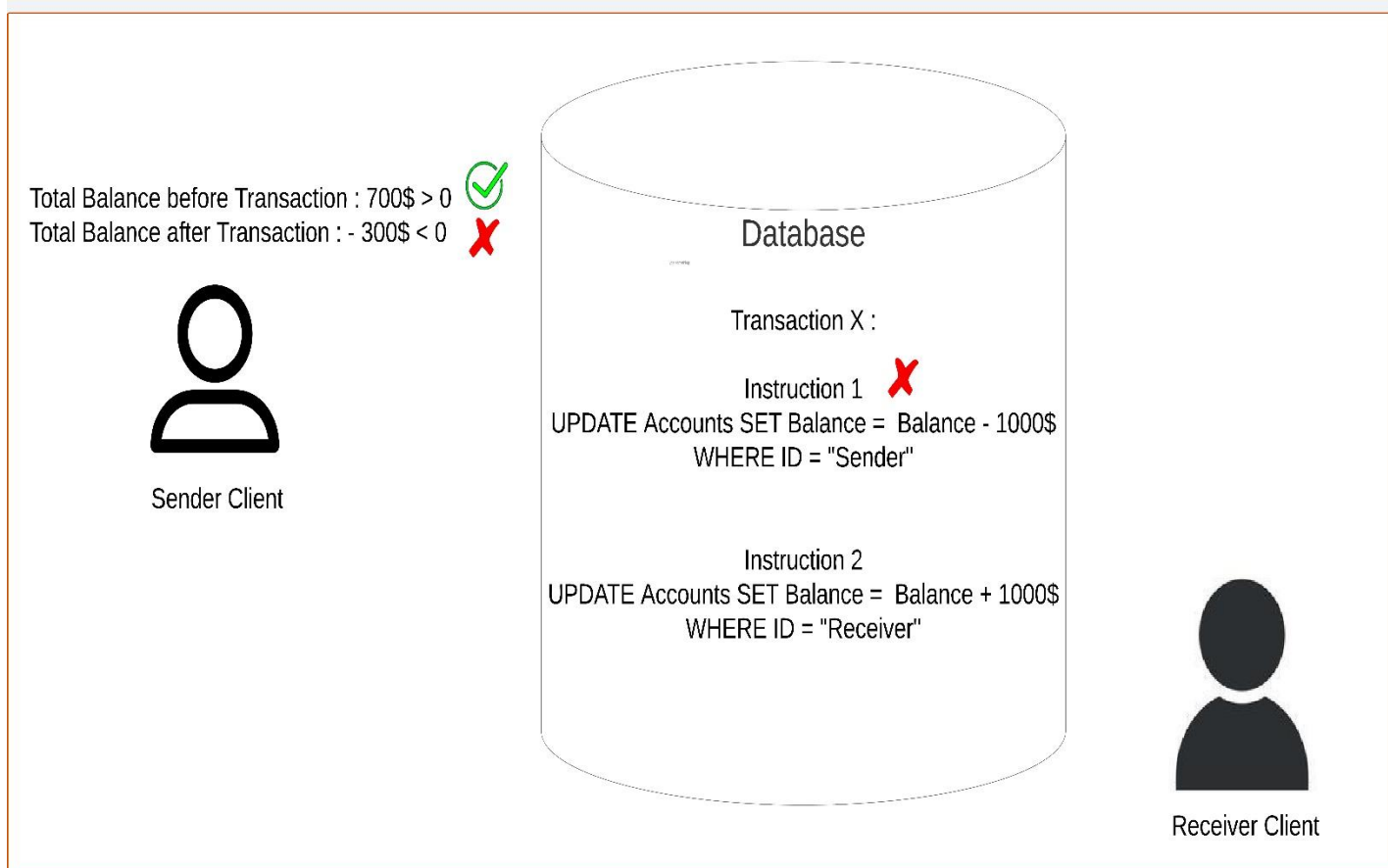
we have a defined rule or constraint on balance column in accounts table :

```
CREATE TABLE Accounts (  
  Col 1 ,  
  Col 2 ,  
  Balance Decimal ,  
  CONSTRAINT Balance_Positive CHECK (Balance >= 0)  
);
```

Case 1 :

when the client try to send money that greater than his total balance this transaction must be failure.

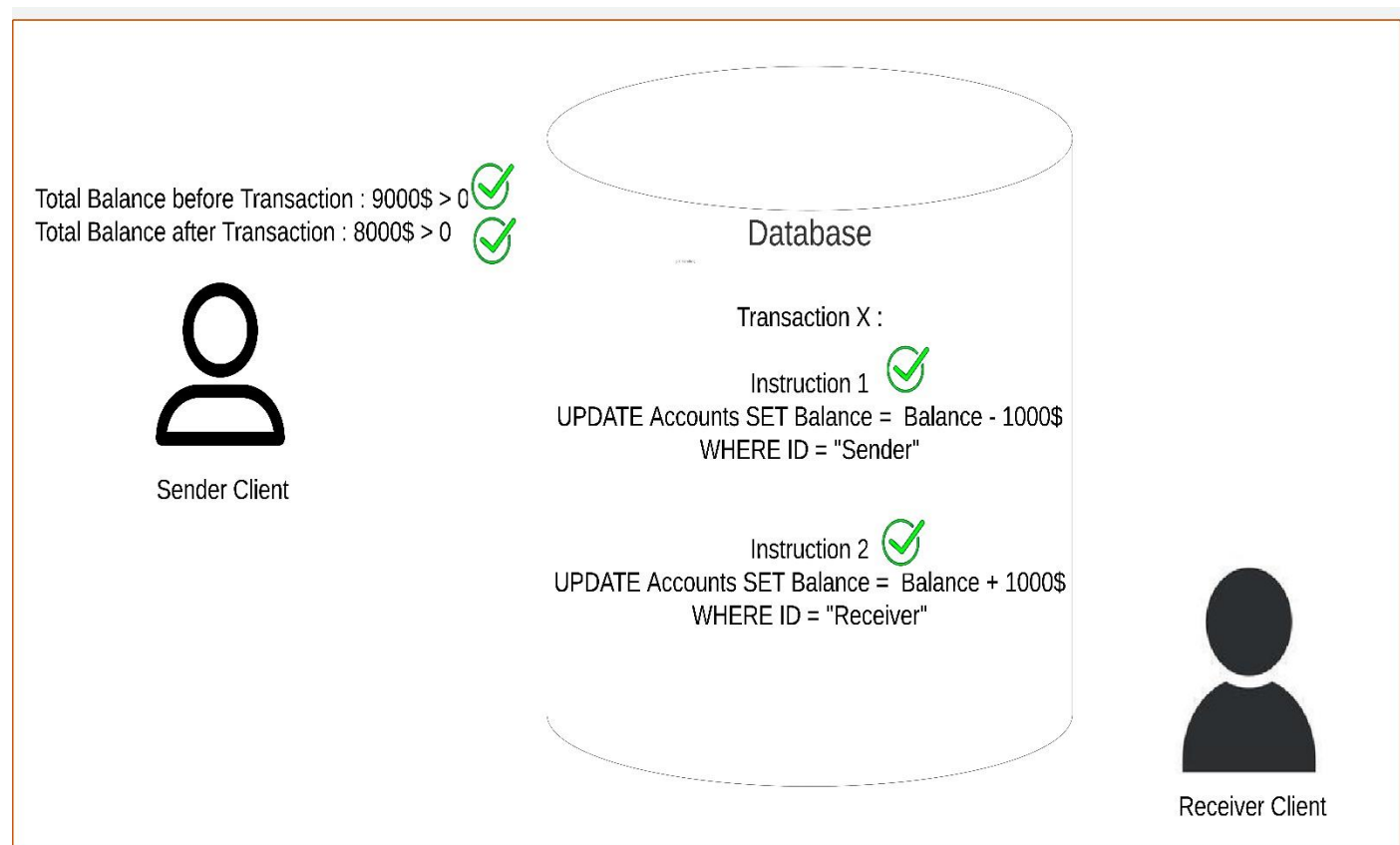
for example client has total balance 700\$ and try to send 1000\$ to other client so this transaction must be failure because it will be move the database from valid consistency state to other invalid consistency state.



Case 2 :

when the client try to send money that less than or equal than his total balance this transaction must be success.

for example client has total balance 9000\$ and try to send 1000\$ to other client so this transaction must be success because it will be move the database from valid consistency state to other valid consistency state.



Consistency vs Atomicity :

One of the misunderstanding concepts I see in the database world is consistency , what is actually mean by consistency ? what is the difference between consistency and atomicity ?

Atomicity :

Atomicity about all instructions or statements is executed successfully or rollbacked (not executed)

Consistency in ACID :

Consistency in ACID about the **results** of this instructions are correct or not.

Consistency in CAP :

Consistency in CAP about if the end user will see the last result or updated value from the replicated node in distributed systems when there are multiple copy of the same data in different nodes , there are different types of consistency in CAP That will be discuss in details in the later sections.

i → (isolation) :

What is isolation ?

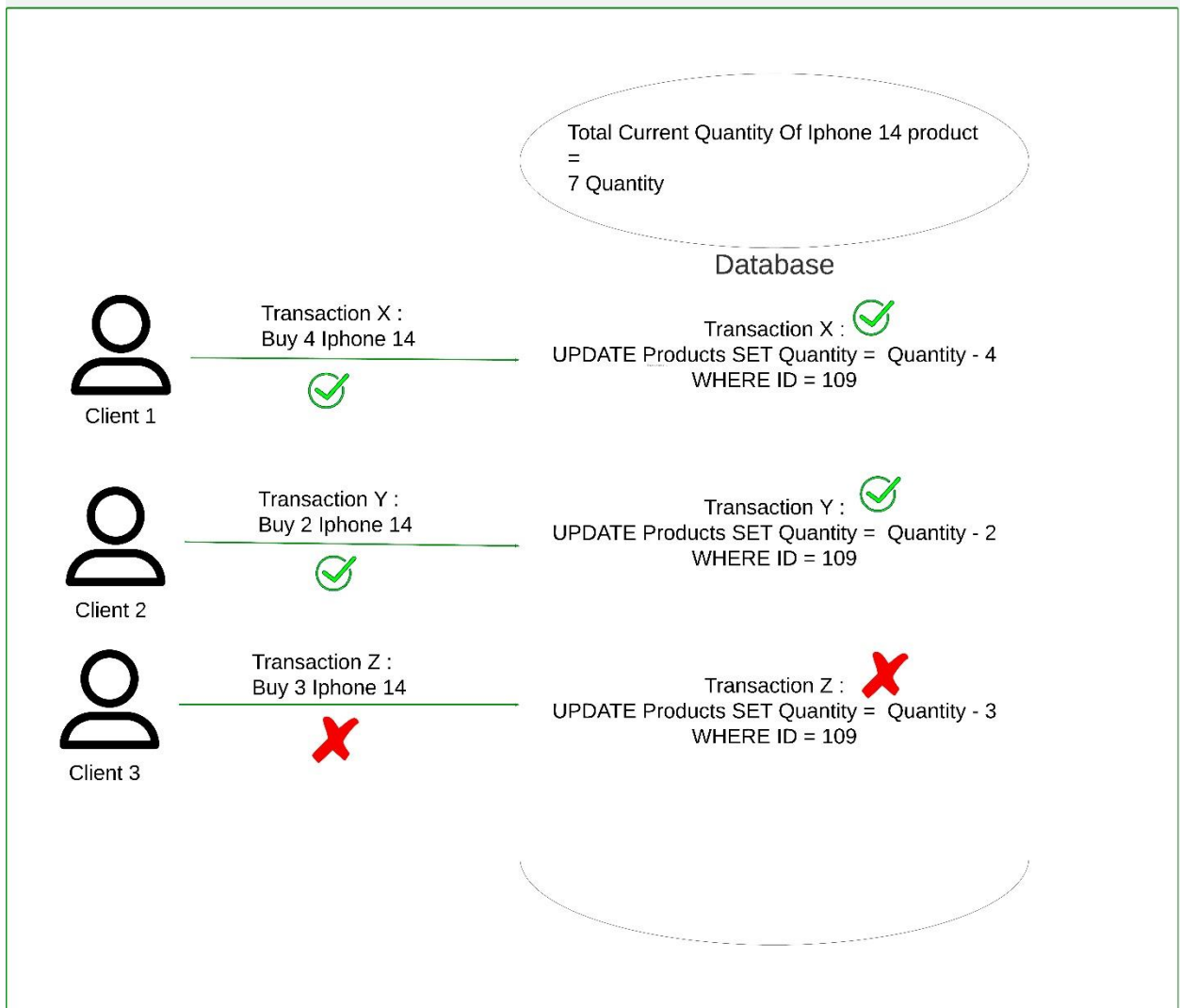
Isolation mean the ability of database to execute concurrent or parallel transactions on the same data as they executed serially without any conflicts.

There are different isolations levels where every level solve some concurrency issues , developers can choose the best one that suitable for their business needs.

We will not discuss the isolation levels here but we will just discuss the general idea of the isolation.

Isolation example :

Three clients try to buy the same product from the stock at the same time.



D → (Durability) :

What is Durability ?

Durability mean the database must be store the transactions and data on non-volatile storage like disk before it mark it as success or return success message to client .

What is the importance of durability ?

if the system or database crash due to power failures or other reason and then restart to working again , so the effects of the transactions will never lose and database continue to work from latest consistency state.

No SQL Databases

What is actually mean by No SQL Database ?

- It just refers to **NOT ONLY SQL**
 - It refer to any database that **does not** apply SQL Database standard on :
 - 1- **architecture level :**
does not store data on tabular format.
 - 2- **implementation level :**
does not implement all or some SQL database standard properties Like Relations , ACID , SQL query language standard.
- So if **any one of the two previous condition** occur the database will be categorized as **No SQL Database**.

Important note 1 :

there are type of No SQL That store data on tabular format but does not implement all or some SQL database standard properties , so this type is categorized as NO SQL Database.

Example : Column Family Database

Important note 2 :

some types of No SQL Database can support SQL Database Standard Properties but in customized or partial cases , so it categorized as NO SQL Database.

Example : some NO SQL Databases support ACID

But per object not per transaction or multi-objects.

NO SQL Databases Types :

1. Documents Databases
2. Key-Value Databases
3. Column Family Databases
4. Graph Databases

Important note 3 :

We will discuss every No SQL Database type In details in the next chapters : -

- What are the business cases of every type ?
- What is the architecture of every type ?
- What are the data model of every type ?
- What are the common operations of every type ?

But in the next section we will discuss the difference between SQL VS NO SQL in general regardless of every NO SQL Database type.

SQL VS NO SQL

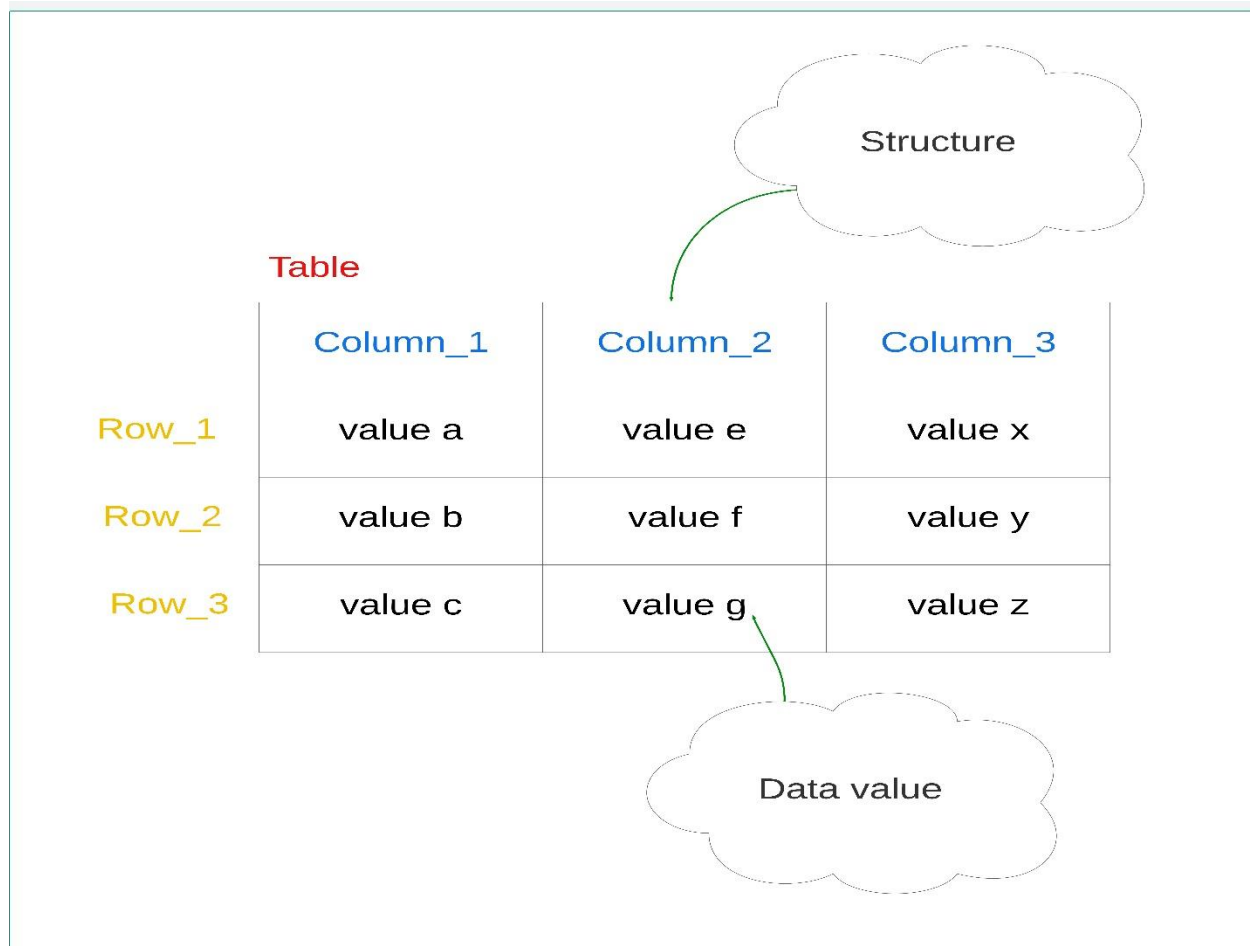
1. Data Models :

What are data model types ?

- 1- Structure data model
- 2- Semi-Structure data model
- 3- Unstructured data model.

Structure data model :

- Every value is stored or mapped to a specific field at a specific location.
- All records of the same type should have the same fields.
- Structure and data values are decoupled or separated from each other.
- The common architecture for this data model type is tabular format (tables ,rows, columns)



Semi-structure data model :

- Semi-structure model is very similar to structure data model but there some differences.

First difference :

Different records of the same types can have different fields.

Second difference :

Structure and values are coupled to each other, structure here is work as meta-data.

- Common patterns : json , xml , html.

Customer_1

Id : 1 ,
Name : Ahmed Khalifa
Birthdate : 5/1/1993
Mobile : 01002245404

Customer_2

Id : 2 ,
Name : Sara adam
Birthdate : 8/8/2000
Email : Sara@example.com

Unstructured data model :

- This data model has not any structure.
- The data is stored in native format as it received.
- Common examples of this pattern are text files or articles , audio files , video files.
- Usually this data model has not query language but use data analysis tools , some of AI/ML algorithms to process this data and extract useful information from it.



How to choose your database based on the data model ?

SQL Databases :

Usually, SQL Databases are suitable for structured data models where the schema is static.

the static schema means all the entities and fields of these entities are known and built by the developers on the development stage.

Business case example :

CRUD Management Systems Like HR Systems.

The screenshot shows a web form titled "AddNewEmployeeByManagerForm_2". The form contains the following fields and controls:

- Employee Name:** A text input field.
- Employee Birthdate:** A date picker showing "Sunday , August 13, 2023".
- Employee Email:** A text input field.
- Employee Phone Number:** A text input field.
- Employee Address:** Three text input fields labeled "Country", "City", and "Street and Building Number".
- Employee Skills:** A dropdown menu with the following options: Database, Full Stack, System Design, Agile, and Distributed Systems.
- Add New Employee:** A button with the text "Add New Employee".

NO SQL Databases :

NO SQL Databases are usually suitable for :

1. Unstructured data models.
2. Semi-structured data models where the scheme is dynamic.

The dynamic scheme mean the attributes of different objects of the same entities are known and inserted in the run time by the end users of the systems.

Business case example :

E-commerce system where every product has different attributes , and this attributes are inserted on the run time by the seller or the admin of company , not by the developers themselves.

Product_1

```
Id : 1 ,
Name : iPhone 14
Category : Mobile
CashPrice : 40000 EG
Attribuities :
{
  Model : 2023
  RAM : 265
  Camera : 48 MP
  Battery : 4000 mAh
}
```

Product_2

```
Id : 2 ,
Name : LG Fully automatic
Category : Washing machine
CashPrice : 12000 EG
InstallmentsPrice : 13200 EG
InstallmentValuePerMonth : 1100 EG
Attribuities :
{
  Model : 2022
  Color : Silver
  Capcity : 8 KG
  Cycle : Spin , Drain
  Dimensions : 60D x 60W x 85H centimeters
}
```

