Temasek Junior College

2024 JC2 H2 Computing

NoSQL Databases 1 – NoSQL Database Management Systems

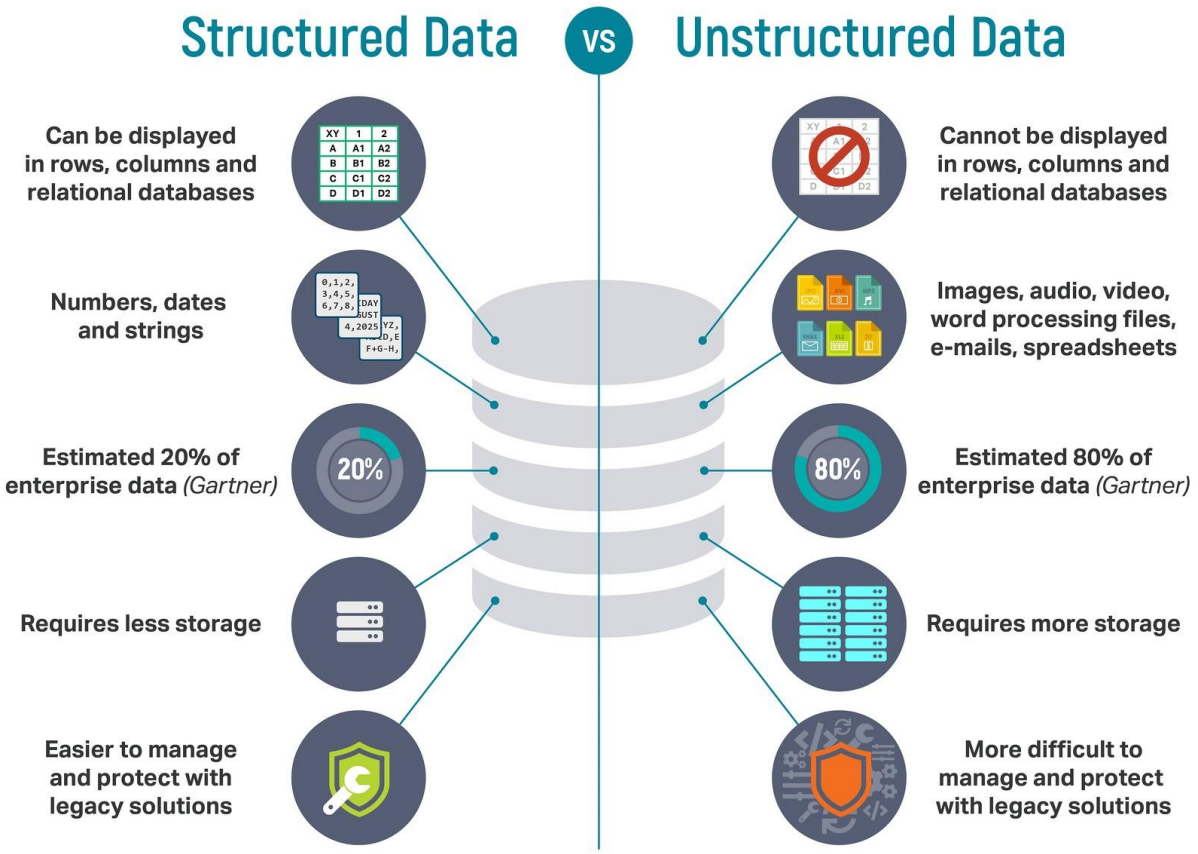
Section	3	Data and Information
Unit	3.3	Databases and Data Management
Objectives	3.3.6	Understand how NoSQL database management system addresses the shortcomings of relational database management system (SQL).
	3.3.7	Explain the applications of SQL and NoSQL.

1 Introduction to NoSQL Databases

1.1 The Need for NoSQL Databases

Relational databases (SQL databases) work well with structured data since each table's **schema** i.e. the precise description of the data to be stored and the relationships between them, is always clearly defined.

However, with the increasing number of ways to gather and generate data, we often need to deal with unstructured data.



Consider the following scenario:

An online apparel store uses a relational database management system (RDBMS) to manage the customers records, clothing inventory and sales data.

A possible entity-relationship model of the system is as follows:

To attract more and retain existing customers, the owners want to include the following:

- *a rating and review feature to allow customers to rate and review their purchases*
- *an exclusive members' zone that*
  - *allows members to create a unique fashion profile and fashion wall*
  - *add other members who share similar fashion interests as friends*
  - *post on the fashion walls of friends*
  - *like the posts of their friends*

In the above scenario, new data needs to be stored. However the data can come in many different formats, be vastly unstructured, and may not be available for all customers.

Storing all this data in the same relational database may not be easy. Even if the owners were intending to do so, they would need to insert new columns to existing tables, add new tables into the database, re-establish the relationships between the tables and so on…

In this case, non-relational databases, also referred to as **NoSQL databases**, can offer a better choice.

1.2 Types of NoSQL Databases

There are four main types of NoSQL databases:

- key-value databases;
- document databases;
- wide-column databases;
- graph databases;

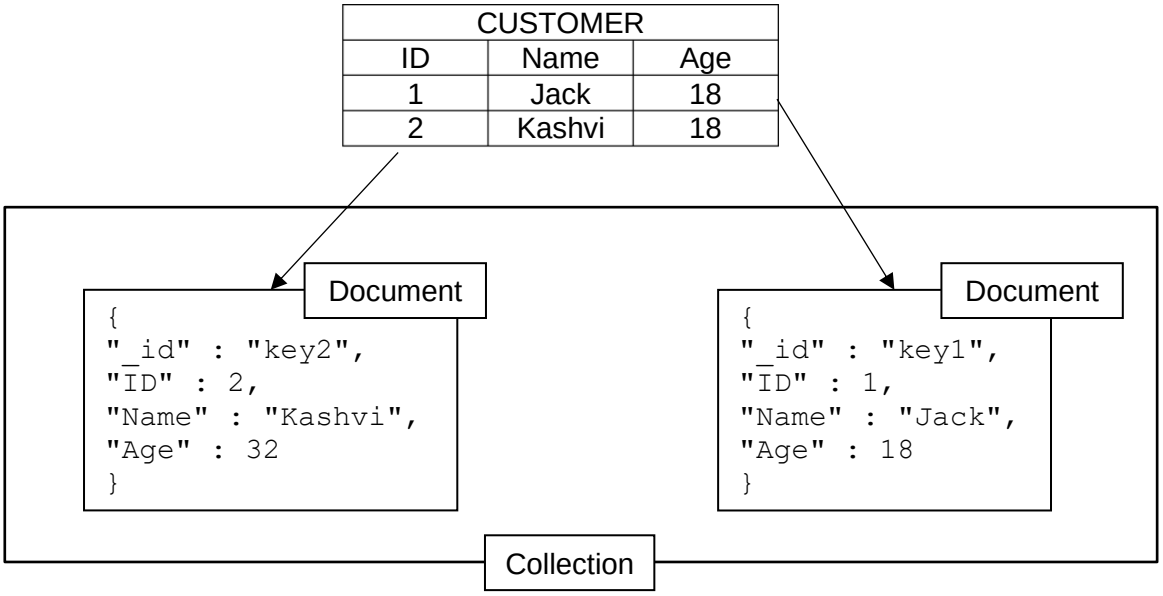
In H2 Computing, we shall focus on MongoDB, a type of document database.

1.3 Document Databases

Document databases work like a hash table, but each key can point to an embedded key-value structure, also known as a **document**, instead of just a single value or data item. (Recall that in a hash table, each key points to a single value or data item.)

Such documents are in the JavaScript Object Notation (JSON) format. (Do not be alarmed. You do not need to know JavaScript to handle the JSON documents in MongoDB.) A JSON document resembles the way dictionaries are written in Python.

The diagram below is a schematic illustration of the structure of a document database in comparison with a relational table.



In a NoSQL document database:

- a **collection** is equivalent to a relational **table** in a SQL database.
- a **document** is equivalent to a **row** of a relational table in a SQL database.
- a **field** is equivalent to a **column** of a relational table in a SQL database.
- an **index** refers to the same value as what an **index** in a SQL database refers to.

In summary:

NoSQL Document Database Terminology	Equivalent SQL Database Terminology
Database	Database
Index	Index
Collection	Table
Document	Row
Field	Column

1.4 Differences between NoSQL Document Databases and SQL Databases

The table below summarises the key differences between NoSQL document databases and SQL databases.

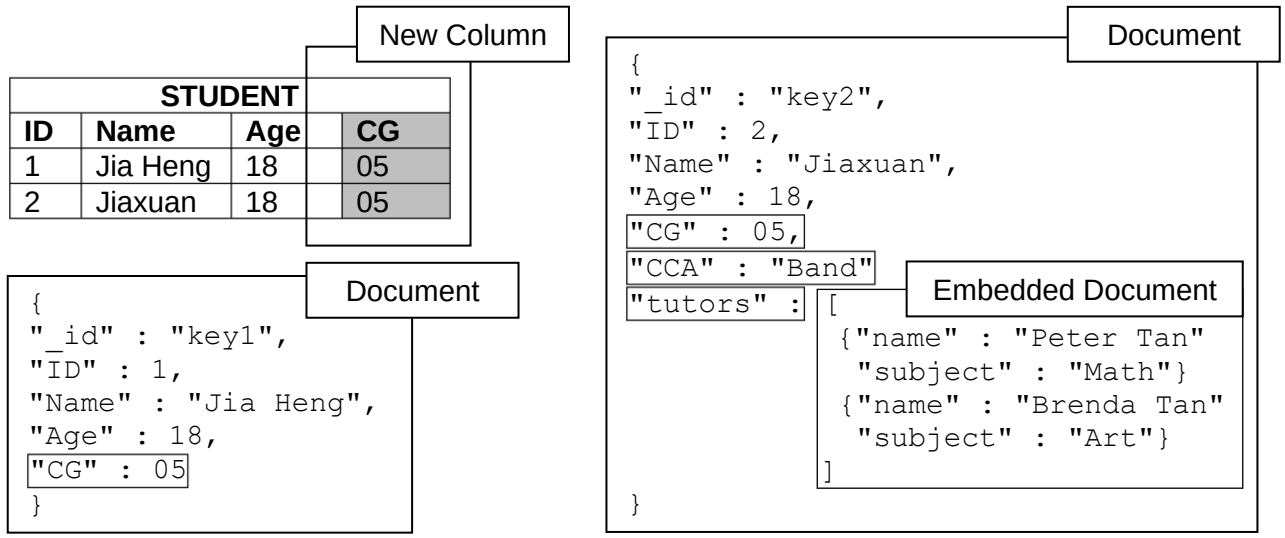
	SQL Databases	NoSQL Document Databases
Data Model (Schema)	• Relational tables followed a rigid, fixed and pre-defined schema	• Flexible, has no pre-defined schema, hence dynamic and can change easily.
Database Structure	• Contains relational tables. • Data stored in the rows of these tables. • Data types of each field in the table is fixed.	• Contains collections. • Data stored as collections of documents. • Flexible in the use of different data types.
Data Relationships	• Relationships between the tables are established using foreign keys.	• Relationships between collections are established via referencing and embedding.
Performing Queries	• Joins are usually performed to obtain data across tables, hence easier for complex queries.	• Can only query one collection at a time, hence limited in query capabilities.
Data Storage in Server	• Data stored in a single server.	• Provides sharding (data spread across different servers).

1.5 Addressing Shortcomings of Relational Databases with NoSQL Databases

1.5.1 Higher Flexibility / Adaptability

- Relational databases have a predefined schema that is difficult (and can be costly) to change. To add a field to a small number of records, the field for the entire table needs to be included. Hence, it is difficult to support processing of unstructured data using relational databases.

- NoSQL databases has flexible schema, hence able to accommodate changes in database requirements. It is thus suitable for storing and processing unstructured data, and is more natural compared to SQL databases.



- Notice that the document for the two records `ID2` has been extended differently.

1.5.2 Lower Storage Costs

- Unlike NoSQL databases, relational databases do not usually support **hierarchical data storage**, where less frequently used data is moved to cheaper, slower storage devices.
- This means that the cost of storing data in a relational database is more expensive than storing the same amount of data in a NoSQL database.

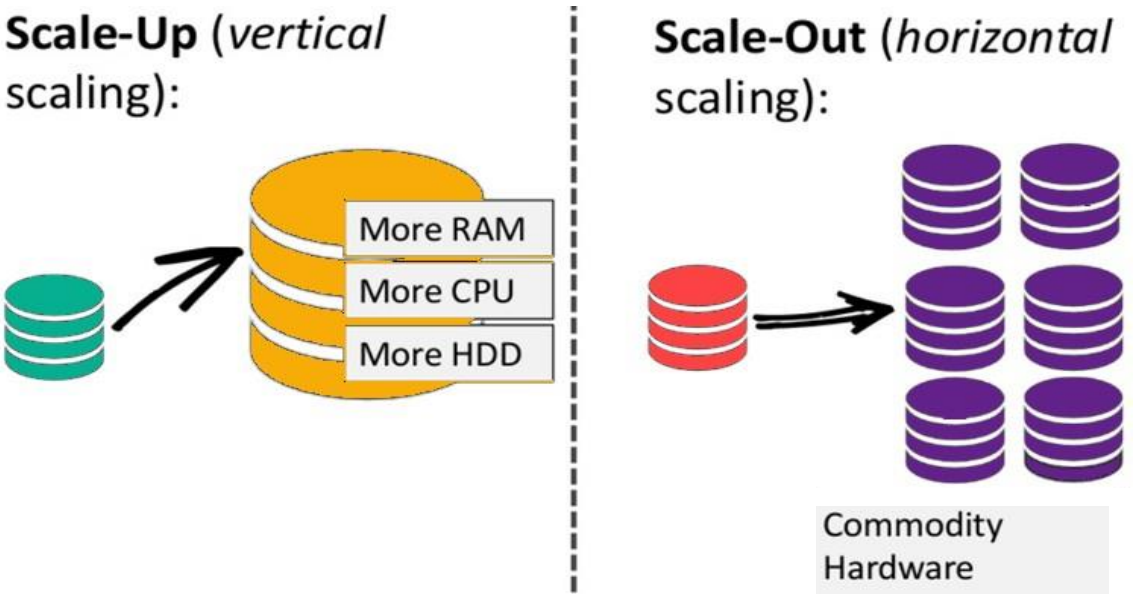
1.5.3 Better Performance Scalability

- Relational databases are mainly vertically scalable while NoSQL databases are mainly horizontally scalable.

- **Vertically scalable** means that improving the performance of a relational database server usually requires upgrading an existing server with faster processors and more memory.
  - High-performance components can be expensive and upgrades are limited by the capacity of a single machine.

- **Horizontally scalable** means that the performance of a NoSQL database can be improved by simply increasing the number of servers.
  - This is relatively cheaper as mass-produced average-performance computers are easily available at low prices.

Relational Databases **NoSQL Databases**



1.5.4 High Availability

- Relational databases are stored in a server, which makes the database unavailable when the server fails (down time).
- NoSQL databases are designed to take advantage of multiple servers so that if one server fails, the other servers can continue to support applications (100% up time). Hence availability is higher in comparison to relational databases.

1.6 Applications of SQL and NoSQL Databases

The choice of whether to use a SQL or NoSQL database depends on the type of data being stored as well as the nature of tasks that the database is required to perform.

SQL databases should be used if:

- The data being stored has a fixed schema.
- Complex and varied queries will be frequently performed.
- The atomicity, consistency, isolation and durability (ACID) properties are critical to the database.
- There will be a high number of simultaneous transactions.

NoSQL databases should be used if:

- The data being stored has a dynamic schema, (i.e., unstructured data with flexible data types).
- Data storage needs to be performed quickly.
- There will be an extremely large amount of data (i.e., Big Data for real time analytics).
- ACID properties are not critical.

2 Designing Schema in NoSQL Document Databases

The following are the rules-of-thumb in the design of NoSQL Document Databases:

- Always design according to user requirement.
- Objects which you want to use together should be combined into one document.
- Otherwise, they should be separated.
- Optimize schema for more frequently use cases.
- Model relationships:
  - Embedded
  - Reference

2.1 Embedded Documents

One to One

```
{
    "_id" : "key1",
    "ID" : 1,
    "Name" : "Lee Young",
    "Age" : 18,
    "Address" : {
                  "avenue" : "Avenue 9",
                  "road" : "Tampines",
                  "country": "Singapore"
                }
}
```

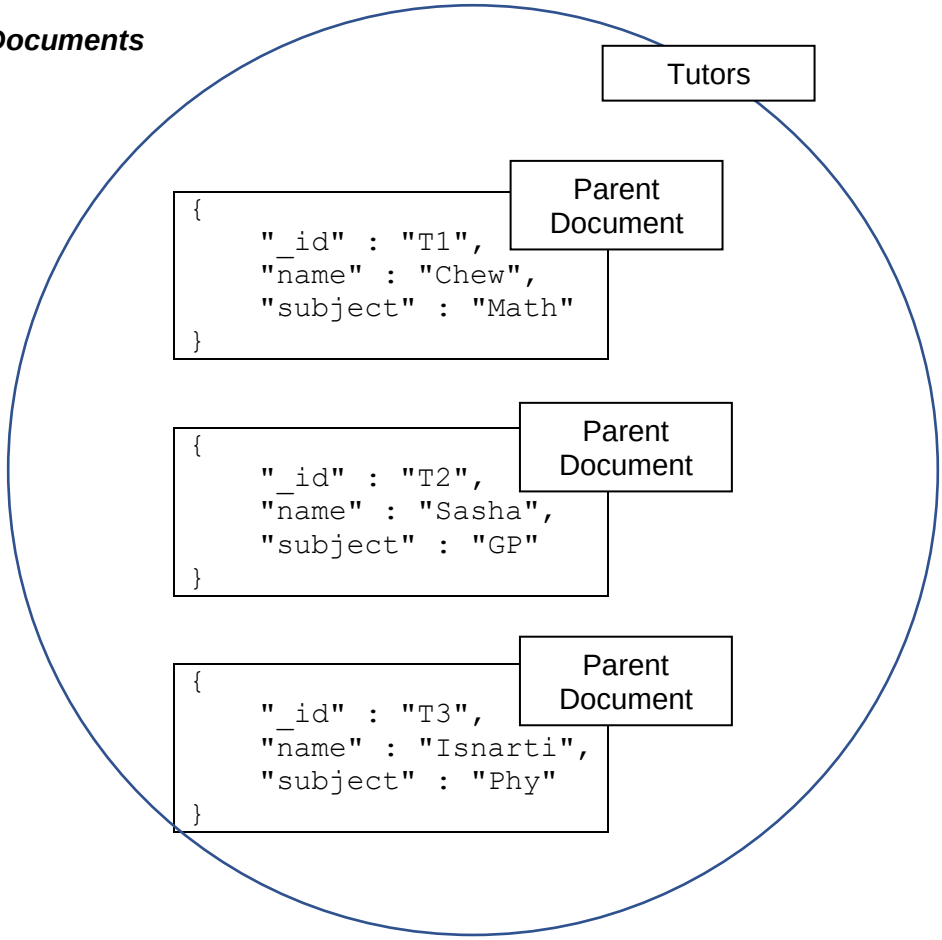
One to Many

```
{
     "_id" : "key2",
     "ID": 2,
     "Name": "Samyak",
     "Age" : 18,
     "CG" : "05",
     "CCA" : "debate",
     "tutors" : [
                  {"name" : "Chew",
                   "subject" : "Math"},

                  {"name" : "Sasha",
                   "subject" : "GP"},

                  {"name" : "Isnarti",
                   "subject" : "Phy"},
                ]
}
```

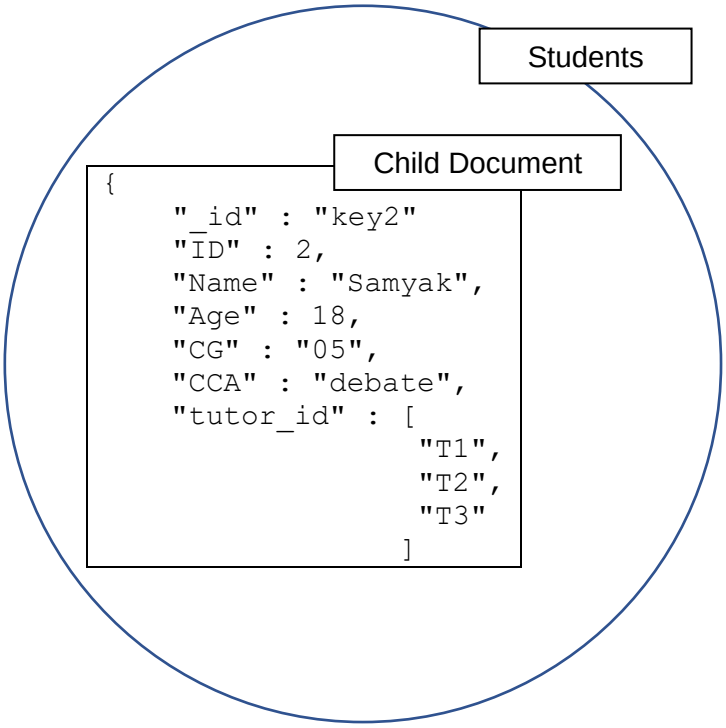
2.2 Document Referenced Relationships

One to Many Relationship Using Reference

(A) Parent Documents



(B) Child Document



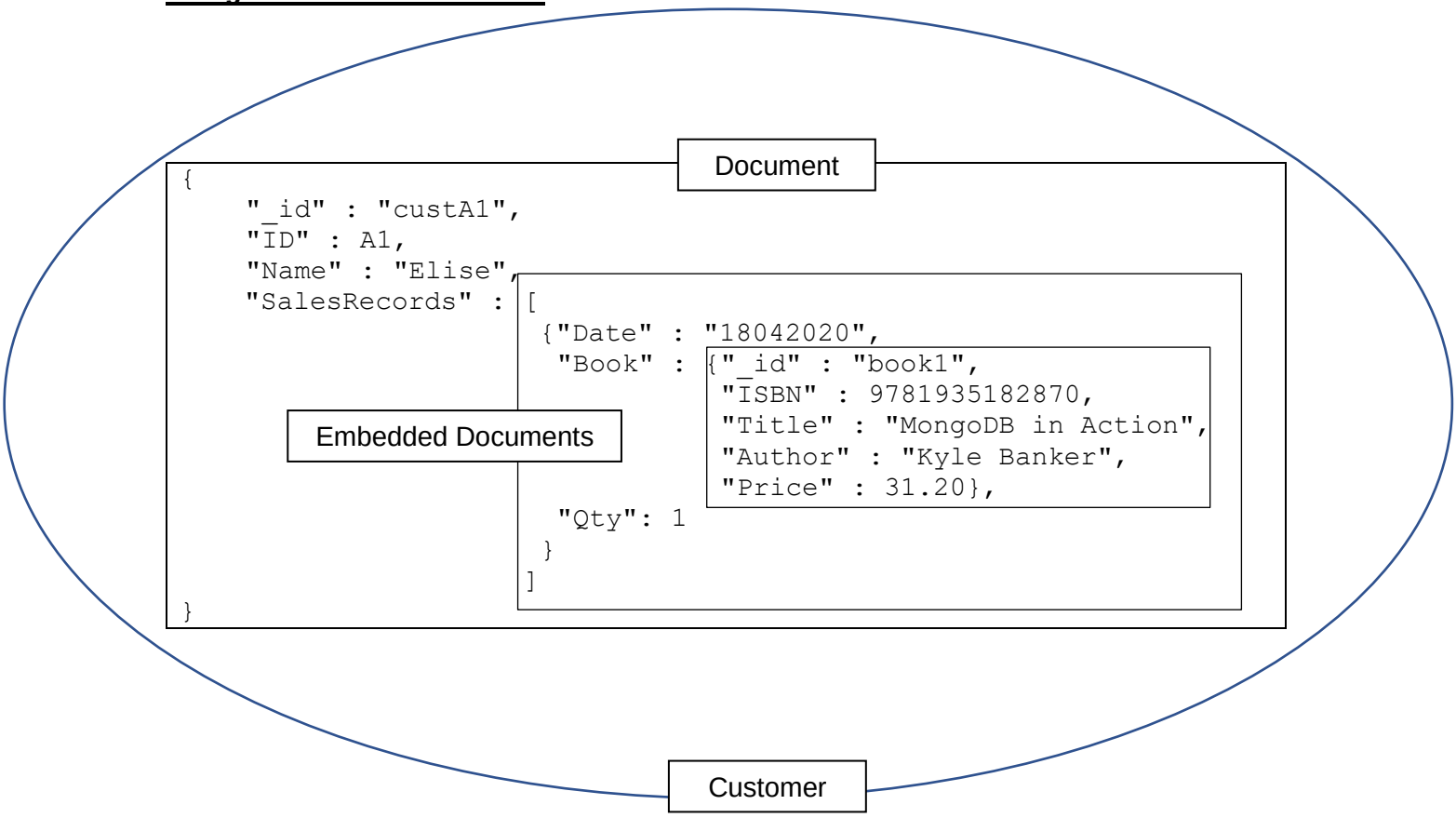
2.3 Relational Database to NoSQL Document Database

Customer	
ID	**Name**
A1	Elise

Sales			
CID*	**ISBN***	**Qty**	**Date**
A1	9781935182870	1	18042020

Book			
ISBN	**Title**	**Author**	**Price**
9781935182870	MongoDB in Action	Kyle Banker	31.20

Using Embedded Documents



Using References