International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 04 | April - 2023 Impact Factor: 8.176 ISSN: 2582-3930

CRS (Computer Reservation Systems) like Ticketmaster shifting to nuanced Data Models and NoSQL Architectures

Shreyansh Padarha[1]

[1] *Department of Data Science, CHRIST (Deemed To Be) University Pune Lavasa*

Abstract - This paper examines the paradigm shift from RDBMS (Relational Database Management Systems) to NoSQL (Not Only SQL) databases in the context of Computer Reservation Systems (CRS) such as Ticketmaster. CRS are systems that handle travel-related service bookings such as airline tickets, car rentals, and hotel reservations. Ticketmaster, a leading company in the online ticket booking industry, has adopted NoSQL databases and other distributed database solutions to improve its platform's scalability, flexibility, and performance. This paper discusses the advantages of NoSQL databases over RDBMS, specifically in the areas of scalability, flexibility, performance, and cost-effectiveness. Furthermore, this paper proposes a wide-column store database as the most suitable type of NoSQL database for a CRS such as Ticketmaster. The advantages of a wide-column store database include horizontal scaling, schema-free data modelling, high-performance read-and-write operations, and cost-effectiveness.

Keywords: Computer Reservation Systems, Ticketmaster, RDBMS, NoSQL, wide-column store database, scalability, flexibility, performance, cost-effectiveness.

INTRODUCTION

The rapid growth of digital transformation has led to an explosion of data in various industries, including the travel and hospitality industry. Computer Reservation Systems (CRS) such as Ticketmaster have emerged as a critical component of this industry, handling the booking of travel-related services such as airline tickets, car rentals, and hotel reservations.

These systems require a robust and scalable database infrastructure to handle large volumes of data, ensuring seamless transaction processing and quick response time to customer queries. Traditionally, Relational Database Management Systems (RDBMS) have been used to manage data in CRS applications. However, as the volume of data grew, the complexities of queries increased, and the need for scalability and performance became more critical, RDBMS databases became insufficient to meet the modern needs of CRS applications.

This paper explores the paradigm shift from RDBMS to NoSQL databases in the context of CRS applications such as Ticketmaster. Ticketmaster is a leading online ticket booking company that has adopted NoSQL databases and other distributed database solutions to improve the scalability, flexibility, and performance of its platform. We discuss the advantages of NoSQL databases over RDBMS, specifically in the areas of scalability, flexibility, performance, and cost-effectiveness.

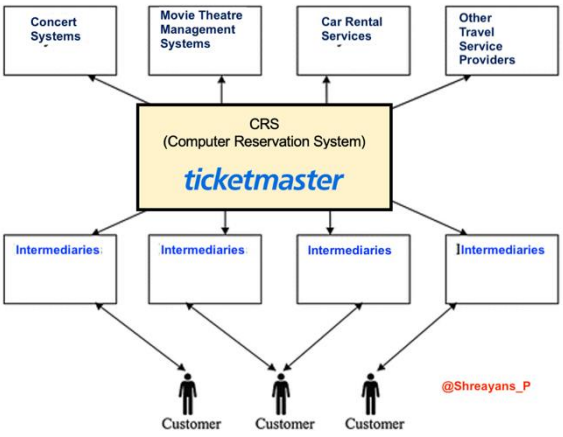
Furthermore, this paper proposes a wide-column store database as the most suitable type of NoSQL database for a CRS such as Ticketmaster. The advantages of a wide-column store database include horizontal scaling, schema-free data modelling, high-performance read-and-write operations, and cost-effectiveness.

PREAMBLE: CRS (COMPUTER RESERVATION SYSTEM AND TICKETMASTER

CRS or Computer Reservation Systems are systems used to handle and manage the booking (scheduling) of travel-related services such as airline tickets, car rentals and hotel reservations. Travel agencies, airlines and other tourism companies make use of CRS. Online ticket booking websites can be considered the digital equivalent of CRS. These websites provide users (clients) a platform to book tickets in advance or in a hurry for events such as concerts, movie shows, theatre performances and sports games.

Ticketmaster is a leading company in the online ticket booking industry. This can be substantiated by the fact that pre covid-19 they sold over half a billion tickets worldwide[1]. This success or sheer dominance has been prevalent for decades. Since 1995, Ticketmaster has maintained a market share of 80% in the United States of America. The company provides a user-friendly platform for purchasing event tickets. One logistical and strategic advantage that has helped them propel themselves is, having partnerships with event organizers and venues, making them the one-stop-shop for event ticketing. The platform (website) also offers features such as event notifications, seat maps and ticket reselling. Ticketmaster has moved towards and adopted new technologies to improve their platform's scalability, flexibility, and performance, including NoSQL databases and other distributed database solutions.

The figure below, created by me shows a depiction of how generally CRS are structured and designed. (fig1)



(fig 1)

In terms of data storage, both CRS and online ticket booking systems require intricate backend infrastructure to manage large volumes of transactions and data. The complexities put forth by modern computer reservation systems can't be handled by traditional RDBMS alone. RDBMS is considered substandard when it comes to horizontal scaling, this hinders the handling of large volumes of transactions and data involved in modern CRSs.

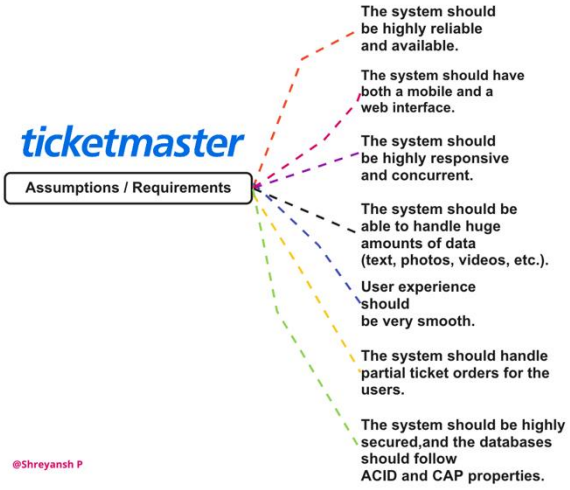
RDBMS INAPPOSITE FOR CONTEMPORARY DIGITAL CRSS

CRS and Online ticket booking systems like ticketmaster, deem managing large volumes of unstructured and semi-structured data pivotal. This data is in the form of user information, event details and transactional data. These systems require a scalable, flexible and high-performance database backend to store the aforementioned data effectively. The storage part is just one component, for a company like ticketmaster time to market is a key aspect of their business model. Whether a customer (purchaser) buys tickets from their website or one of their competitors, is heavily dependent on the query response rate. This prompts the retrieval rate of data, upon querying to be high.

Ticketmaster's switch to hybrid data modelling

Initially, Ticketmaster chose RDBMS for its data modelling because it was the most popular and widely used database technology. RDBMS was specifically designed for handling structured data, which made it apt for managing transactional data related to ticket sales. Additionally, RDBMS offers data integrity, consistency, and security, which is critical for Ticketmaster's processes. However, as the volume of data rose, and the complexities of queries grew, ticketmaster realized that RDBMS is unsuitable for modern tasks. As a result, it becomes difficult to scale RDBMS databases to handle large volumes of data. Moreover, RDBMS databases are expensive to maintain and require significant investment in both hardware and software.

To better understand the technical and business requirements of ticketmaster or any CRS for that sake, I have made some assumptions of (fig2). These play a key role in developing an efficient and profitable website/app.



(fig 2)

Characteristics shown by NoSQL Databases

The requirements mentioned in fig1 (2.1) clearly dictated the need for a paradigm shift from RDBMS to a more flexible and seamless data modelling approach. This is where NoSQL databases came in. NoSQL databases offer some key advantages over RDBMS (Table 1).

Advantages of NoSQL over RDBMS	
Scalability	NoSQL databases are designed to handle large amounts of unstructured and semi-structured data, making them ideal for Ticketmaster like big data applications. NoSQL databases can scale horizontally. This enables adding nodes to the database cluster as and when needed.
Flexibility	NoSQL databases are schema-free, this creates a foundation for varied unstructured data to be stored without creating a pre-defined schema. This makes NoSQL databases

	ideal for CRS applications like ticketmaster where data structures constantly change.
Performance	NoSQL databases are optimized for read-and-write operations, which makes them ideal for high-performance and velocity applications. Along with that, NoSQL databases can be designed, customised to be highly available, meaning that they can continue to operate even if one or more nodes fail
Cost-effective	NoSQL databases are typically open-source and do not require expensive hardware or software licenses. Also, as mentioned above, NoSQL databases can be scaled horizontally, without requiring expensive hardware.

CONDITIONING THE SUITABILITY OF NOSQL DATABASES FOR CRS

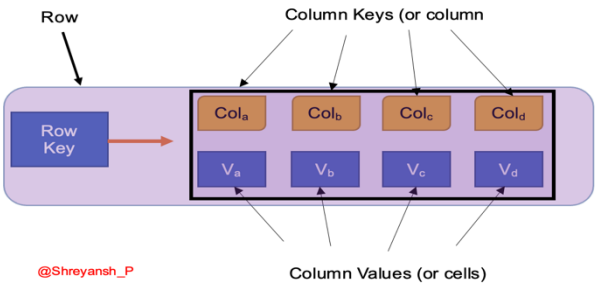
Ticketmaster is accustomed to handling millions of transactions a day. To handle this scale of data, Ticketmaster needs a database backend that is scalable, flexible and high performing. While various types of NoSQL databases exist, the most suitable type for a Computer Reservation system like Ticketmaster is a wide-column store database.

Wide-column store database and its advantages

A wide-column store database, also known as a column family database, is designed to handle large volumes of unstructured and semi-structured data. This type of database uses a flexible schema model, allowing users to store data with different attributes and data types. Wide-column store databases can also scale horizontally by adding more nodes.

Understanding the fundamentals of wide-column databases are important to correctly interpret their role in aiding Ticketmaster's website (fig 3). In a wide-column NoSQL database, the data is organised as columns and rows similar to a table in RDBMS. However, unlike relational databases, the column can vary from row to row, aiding flexibility. A unique element of the wide-column database is its ability to store large amounts of data while handling high throughput and low latency requirements. These databases are designed to manage a write-heavy workload, making them ideal for applications like Ticketmaster that require fast and efficient data storage and retrieval.

The key difference between wide-column NoSQL databases and other NoSQL databases is the structure of the data model. While other NoSQL databases are stored in simple key-value pairs. But, wide-column NoSQL databases offer a more structured data storage solution.



(fig 3)

Ticketmaster's adoption of ScyllaDB over Apache Casandra

Upon doing a comparative analysis of ScyllaDb, Apache Casandra and DataStax Enterprise, Ticketmaster decided to go with ScyllaDB for its Application and logistics data management. Here are some advantages of ScyllaDB, that might overlap with the advantages of NoSQL:

1. **Improved Performance:** ScyllaDb is known for its high performance and low-latency characteristics. Ticketmaster can achieve faster read and write operations, increasing the overall performance of the website.

2. **Scalability:** ScyllaDB is designed to scale horizontally, which is ideal for Ticketmaster because it requires scalable solutions to handle the high volumes of ticket sales and website traffic.

3. **High Availability:** ScyllaDB provides built-in replication and fault tolerance features, ensuring the data remains available even in the face of unexpected node failures.

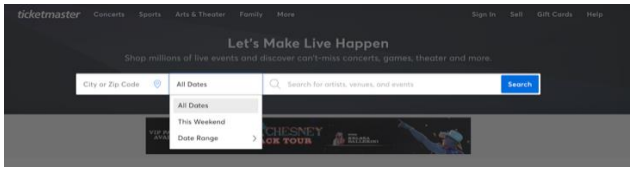
4. **Cost-effectiveness:** ScyllaDB is open-source software, which means that Ticketmaster can avoid costly licensing fees involved with proprietary software(databases).

Elements from https://www.ticketmaster.com/ that dictate the need for wide column DB

Visiting Ticketmaster's website and observing its functionalities, provides evidence that a wide-column store database is the best choice for its needs. For instance, the

website allows users to search for events based on various criteria such as location, event type and date range (fig 3). This type of data requires handling large volumes of unstructured data, as most of this data is either stored on the cloud or retrieved through the API of third-party apps. Additionally, while booking concerts, the high traffic on https://www.ticketmaster.com/ needs to be responded to efficiently and with urgency.

The above-mentioned **(3.1)** benefits of wide-column databases will come into play here.



(Fig 3)

PREEMPTING DISTRIBUTION MODELS USED FOR TICKETMASTER'S WEBSITE APPLICATION

Designing a distribution model for a complex CRS like Ticketmaster is vital to consider several factors including scalability, availability, reliability, and data locality. In the case of a poorly designed distribution model, the possible consequences can be dire for a large-scale company like Ticketmaster. The ramifications include downtime, performance issues and data inconsistencies. Therefore, selecting the right distribution model is imperative to ensure that the system can handle large volumes of data and provide high availability and reliability.

Deciding which distribution model to choose

There are several distribution models to consider, including single server, sharding, master-slave replication, peer-to-peer replication, and combining sharding and replication. Each model has its strengths and weaknesses, and the best option depends on the specific needs of the application. Now we'll try to cover all the possible data distribution models that can be implemented for Ticketmaster (table 2)

Data Distribution Model	Description
Single-Server model	**a)** A single-server model as the name suggests involves running the entire application on a single server. **b)** While the approach is simple and straightforward it's not suited for large-scale applications like ticketmaster. **c)** It can also get overloaded with traffic, resulting in slow response times, downtime and poor user experience.
Sharding	**a)** Sharding is a technique where data is partitioned across multiple servers. Each server stores a subset of the data. **b)** This approach can improve performance and scalability by distributing the workload across multiple servers. **c)** Managing data consistency across different shards can be an added dimension of complexity in sharding. **d)** In a sharding data distribution model, the application must be designed to support partitioning the data into separate shards. **e)**Each shard must be large enough to make the partitioning worthwhile, but small enough to fit into a single server's memory. The application being used must also have a way to map each piece of data to the appropriate server, which can be done using an algorithm. For example, hashing.
Replication	**a)** Replication is a method that involves creating multiple copies of the data and storing them on different servers. **b)** Replication can improve the availability and reliability of data by ensuring that data is always accessible, even if one server fails. **c)** Replication can also increase the complexity of the model or data flow. One such complexity can be managing data consistency between different replicas. In a replication model, changes made to one replica of the server must be propagated and applied to all other replicas to ensure consistency. This can be done using either synchronous or asynchronous replication. This process can be extremely time-consuming, and prone to fault tolerance.
	a) Master-slave replication is a type of replication where one server (the

Master-Slave Replication	master) is designated as the primary copy of the data, and all other servers (the slaves) replicate changes from the master. **b)** Master-slave replication can improve read scalability, as the slaves can handle read queries while the master handles writes. **c)**Replication can also introduce complexity, such as managing failover in case the master fails.
Peer-to-Peer Replication	**a)** Peer-to-peer replication is a type of replication where all servers are equal peers, and changes are propagated between them. **b)** This approach can improve write scalability, as multiple servers can handle writes simultaneously. **c)** However, it can also introduce complexity, such as managing conflicts between replicas.

Table 2

For every data distribution model, there are certain drawbacks that can be detrimental in nature. In such a situation, we can combine two models for optimum performance.

Combining Replication and Sharding

Combining sharding and replication can provide the best of both worlds. By partitioning the data across multiple servers, sharding can improve performance and scalability. Replication can then be used to ensure that each shard has multiple replicas, providing high availability and reliability. This approach can also improve data locality, which is important for a system like Ticketmaster. Data locality refers to the idea that data should be stored close to where it will be used. For example, if a user is searching for events in a specific location, the data related to that location should be stored on servers close to the user. This can reduce latency and improve performance. In the case of Ticketmaster, a combination of sharding and replication would be the most suitable distribution model.

CONCLUSION

The paradigm shift from RDBMS to NoSQL databases is crucial for companies like Ticketmaster which require a scalable, flexible, and high-performance database backend to store large amounts of transactional data. The advantages of NoSQL databases over RDBMS, including scalability, flexibility, performance, and cost-effectiveness, make them the most suitable option for modern computer reservation systems. Ticketmaster has adopted a wide-column store database as a type of NoSQL database, which offers horizontal scaling, schema-free data modelling, high-performance read-and-write operations, and cost-effectiveness. The proposed research highlights the significance of NoSQL databases in the context of CRSs such as Ticketmaster, and the advantages they offer over traditional RDBMS. This research is relevant to technology professionals and researchers in the field of data management, as well as business analysts who seek to improve their platform's performance, scalability, and flexibility.

REFERENCES

[1] Live Nation Entertainment Investor Relations, via U.S. Securities and Exchange Commission. "Form 10-K for the Fiscal Year Ended December 31, 2021," Page 3 (Page 5 of PDF).

[2]Yale Insights. "Did Ticketmaster's Market Dominance Fuel the Chaos for Swifties?"

[3]https://www.oracle.com/in/database/nosql/what-is-nosql/

[4]https://resources.scylladb.com/ecommerce-retail/performance-testing-by-ticketmaster-scylla-vs-cassandra-vs-datastax