Cockroach Labs

CASE STUDY

How Booking.com simplified order management with CockroachDB

Booking.com

Booking.com

How Booking.com simplified order management with CockroachDB

INDUSTRY:
Online Travel

CHALLENGES:
Simplifying their architecture to improve the customer experience.

SOLUTION:
An highly available, always correct, order management system built on CockroachDB.

About Booking.com

Founded in 1996 in Amsterdam, Booking.com is one of the world's leading digital travel agencies. The company's mission is to take the friction out of travel and make it easier for everyone to experience the world. Over the past 15 years, the company has welcomed over 6.8 billion guest arrivals across all properties and has expanded its offerings to include flights, car rentals, public transportation, and tickets to attractions.

A few years ago, Booking.com invested in an initiative to become a one-stop solution for all travel-related needs which they call the "Connected Trip". In the background, this involved the need for building smart systems that would allow for seamless scaling and operations, while also complying with various global regulations. The team at Booking.com needed a distributed SQL database to support this initiative and selected CockroachDB as the new home for its Order Platform.

Watch: How Booking.com simplified order management with CockroachDB ▸

3
REGIONS

99.999%
AVAILABILITY

20TB
OF DATA

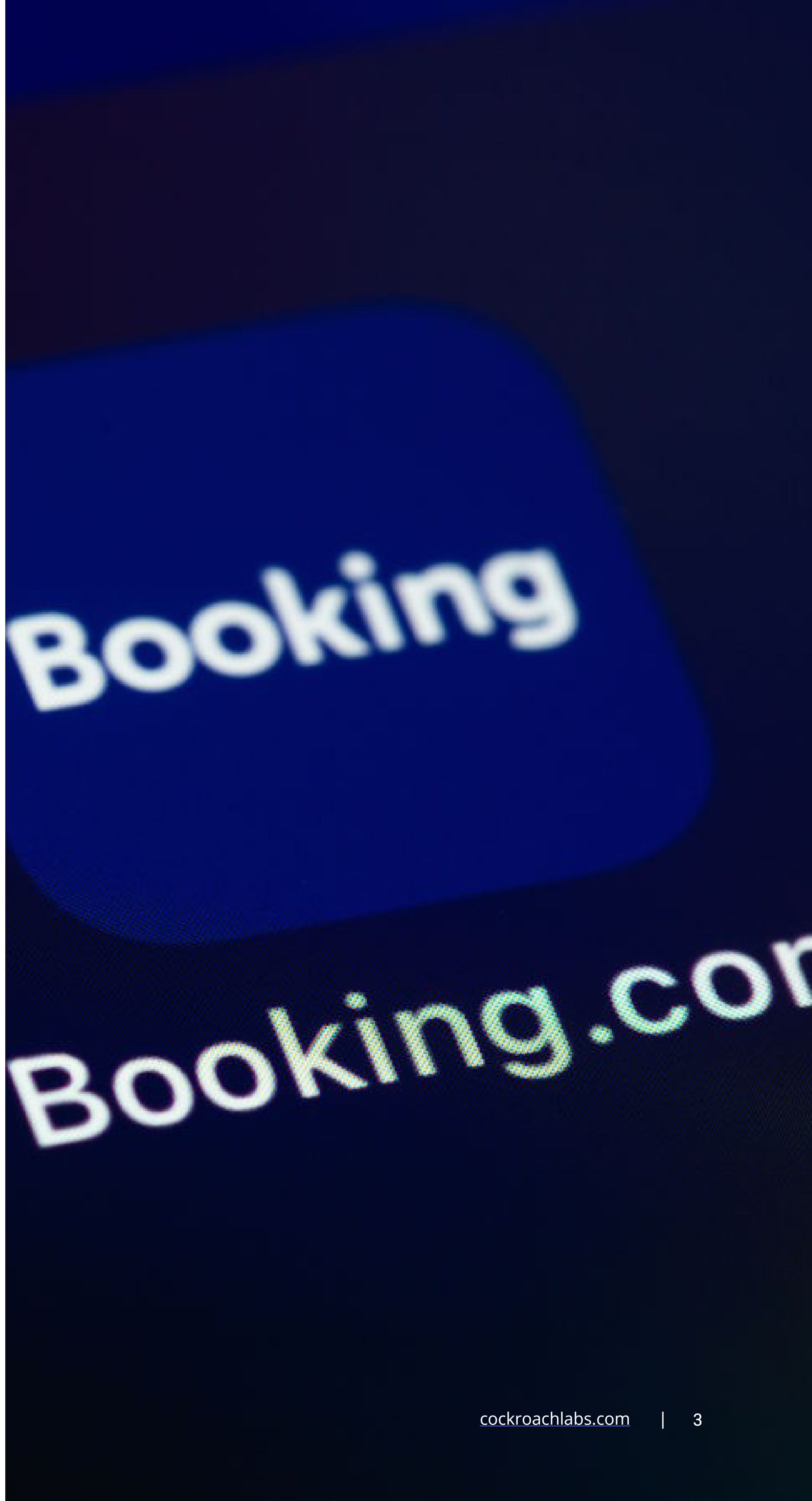
Finding a new destination for reservations

Back when the company was founded almost 30 years ago, there were not a lot of database options available and they ended up adopting MySQL as their primary data store. As the business began to scale and the data set grew, they started facing a lot of scaling challenges. They also had a couple of massive failures that impacted business continuity.

In their effort to deliver the "Connected Trip" they decided to build a centralized system for all of the reservations data instead of the siloed systems. The new system, called Order Platform, was designed on Cassandra. This was primarily because of the lessons they learned over the years with managing large scale sharded MySQL that often lead to complicating the application layers as well as downtimes during mater failovers.

As time went on, they started to face several issues with the NoSQL solution. Reservation data is transactional by nature and Cassandra doesn't have proper support for ACID transactions. This forced the team to use highly denormalized data models to guarantee atomicity. However, this wasn't enough because Cassandra provides only row level isolation which is not suitable for multi-step read-modify-write across multiple rows/partitions in a single operation.

The team implemented a locking system for such cases that relied on lightweight transactions to overcome this limitation, but they soon learned that this wasn't the right solution since sometimes they would timeout (just because of the way it works). It was hard to know if the timeouts were actually because of a legitimate failure due to unfulfilled conditions in the query, or because of another underlying issue.



Also, reservations data was in high demand across the business as it powers tons of use cases. This includes front-end use cases that serve travellers and partners which require fast retrieval, as well as accounting and analytics which require streaming of data. Due to limited support of secondary indexes, they started using OpenSearch to help with this issue. However, that quickly increased operational and maintenance costs quite significantly because of the need to replicate massive datasets across multiple databases, scale them, and guarantee that they stay in sync in near real-time.

Overall, this order management system was becoming too complex and creating cognitive overload, so they started looking at the database landscape for a new solution. The team knew there would be no silver bullet, but wanted less operational overhead without having to compromise on the capabilities that they provide to users.

In summary, their requirements for a new solution included:

- Strong data consistency with ACID guarantees
- Change Data Capture (CDC)
- Support for secondary indexes
- Managed solution to reduce operations
- Compatibility with existing ecosystems of monitoring, security and access management
- SOX compatible and PCI compliant

The team selected three databases that they thought would fit including CockroachDB. During the evaluation process they tested how well these databases could scale, and how easy they were to operate. After completing a proof of concept (POC), CockroachDB was selected as the new solution for Order Platform.

"When looking for a new solution, we didn't want to compromise the capabilities that we provide for our users. We did proof of concepts with three databases and evaluated them against real world scenarios to see how easier they were to scale, how easy they were to operate, how easy they were to integrate into our existing ecosystem, etc. Ultimately CockroachDB was the winner."

Mahmoud Nagib

Principal Software Engineer Booking.com

Migrating an order management system

The migration process from Cassandra to CockroachDB had to be orchestrated slowly and carefully since they had massive amounts of reservation data. Additionally, because Order Platform is under SOX compliance, they had to implement a lot of controls and tools around the migration process to ensure they wouldn't accidentally tamper with the data that was moved to CockroachDB.

Ultimately they successfully migrated Order Platform to CockroachDB which makes up around 20TB of data today.

Here's a glimpse at how the platform works: Once a customer starts the checkout process for any product or tries to modify their booking, the Order Platform becomes responsible for orchestrating different business processes required to fulfill that reservation request. For example, it could start by calling the payment systems to start the payment process. If the payment is approved, it triggers the fulfillment process for the requested products (e.g. flight, hotel, car...etc.)

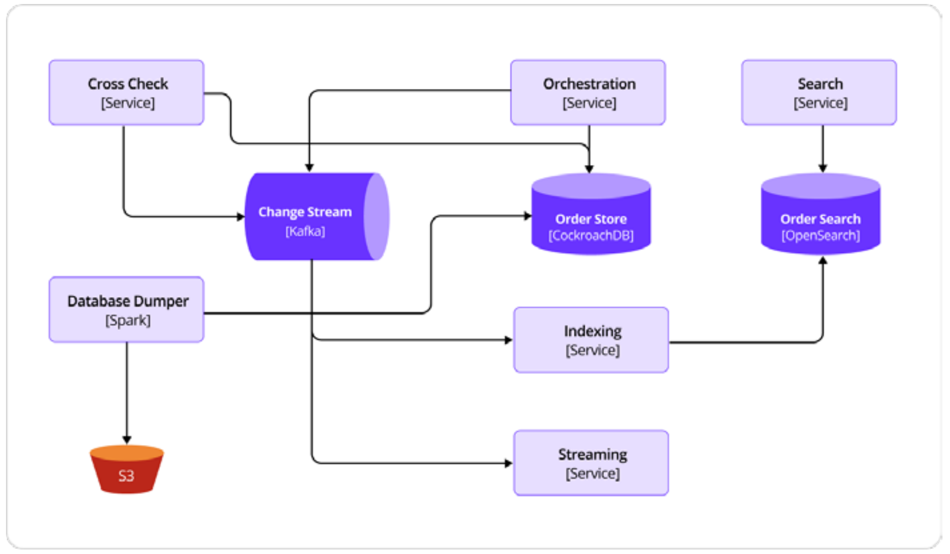
Once the entire process is complete, the reservation data gets stored in the Order Platform which is then used to enable a lot of downstream use cases such as the display for the confirmation page, or used later to retrieve the user's booking. Other use cases within Booking.com such as the ML system (training ML models) and finance systems (billing) rely on this data, making it a critical system that needs to be resilient.

Creating a highly available, simplified architecture

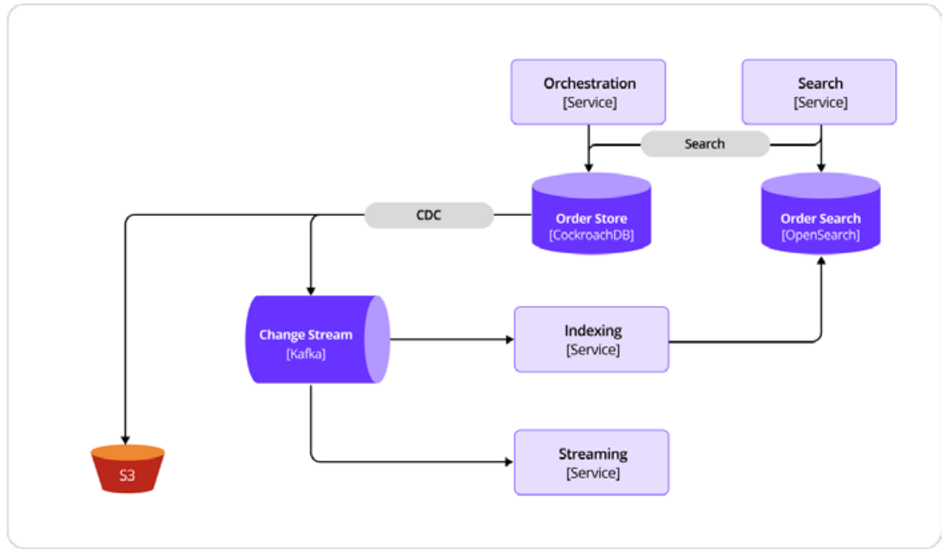
Booking.com was able to simplify their application logic by switching to CockroachDB. Because CockroachDB has native CDC and ACID guarantees, they were able to remove extra cross checks and eliminate the need to maintain those services.

CockroachDB also has secondary indexes which aids with the retrieval of data and is more cost efficient. You can see the before and after illustrations below:

Before:

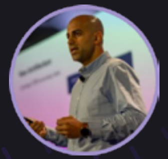


After:



In terms of deployment, Booking.com uses a managed service offering called CockroachDB Cloud. They are running a single cluster across multiple regions – leveraging primarily Frankfurt and London, with Ireland available as a third region with extra compute resources. Running across three regions is a best practice for creating a highly available setup.

"With CockroachDB, building on top of a consistent CDC is a great benefit. Of course resiliency as well. The fact that we can have nodes go down without causing a disturbance is a huge plus. And we know that it's easy to scale CockroachDB when we need to – just add more nodes and it scales horizontally."



Mahmoud Nagib
Principal Software Engineer Booking.com

Advice for new travelers

If you are dealing with a lot of data, Mahmoud recommends that you really look under the hood to understand how CockroachDB operates. He has seen others in the company try to move from MySQL to CockroachDB thinking their features would map one to one, but this is simply not the case.

Mahmoud said it's also wise to think about database tradeoffs. For example, the promises between Cassandra and CockroachDB are different. Cassandra has higher availability for writes, but lacks strong ACID guarantees. On the flip side, CockroachDB provides strong ACID guarantees while delivering a 99.99998% SLI for writes which is still acceptable for their use case.

Because CockroachDB is a distributed SQL database, it's wise to think carefully about the implementation in relation to goals. For example, is it better to put multiple clusters in one region, or to scale a single cluster across multiple regions? Both setups have their performance and resiliency tradeoffs. Or how should you design your schema to get the best performance? At the end of the day, it depends on your use case and Cockroach Labs experts are available for support.

What’s next for Booking.com

The Booking.com team is working to extend CockroachDB as the source of truth for streaming use cases which will rely heavily on CDC. This will allow them to completely move off Cassandra and remove “some of the components that are actually causing havoc.” The team is also working on a second use case connected to Order Platform which is their Basket application.

In terms of where Booking.com is heading, they plan to focus on expanding their reach into the U.S. market. And like any tech-forward company, they are also discovering ways that they can leverage AI to enhance the customer experience. Check out Booking.com’s Tech Blog to learn more.

Ready to get started?

Go hands-on with 100% free CockroachDB Serverless. Spin up your first cluster in just a few clicks.

Get Started →

Compare Products →