Introduction to CockroachDB

2021-02-18

# Introduction to CockroachDB

CockroachDB is an open-source, distributed, transactional, relational SQL database system. That’s quite a mouthful! But one way to look at it is that CockroachDB leverages both the strengths of the last generation of database systems - strong consistency, the power of SQL, and the relational model - and the strengths of modern distributed systems that allow CockroachDB to achieve global scale and continual availability.

To understand the strengths of CockroachDB, it’s worth revisiting the evolution of database systems. We’ll see that CockroachDB is the latest in a succession of technology advances that make it a compelling addition to the database ecosystem.

## A Brief History of Databases

Data storage and data processing is a core feature of human civilization. The earliest written records - dating back 10,000 years - represented agricultural accounting records. These cuneiform records, recorded on clay tablets, are genuinely analogous to the databases that support modern accounting systems such as Xero.



Cuniform table circa 3000BC [[1]](#footnote-1)

However, today we generally use the term database to refer to a collection of information stored using digital computing technology - specifically a DataBase Management System (DBMS).

### Pre-relational Databases

The first digital computers had negligible storage capacities and were used primarily for computation — for instance, the generation of ballistic tables, decryption of codes, and scientific calculation. However, as magnetic tape and disks became mainstream in the 1950s, it became increasingly possible to use computers to store and process volumes of information that would be unwieldy by other means.

Early applications used simple flat files for data storage. But it soon became obvious that the complexities of reliably and efficiently dealing with large amounts of data required a specialized application of its own. Consequently, the first DBMS systems emerged.

Early DBMS systems ran within monolithic mainframe computers, which also were responsible for the application code. The applications were tightly coupled with the database management system and processed data directly using procedural language directives. By the 1970s, two models of database system were vying for dominance - the **Network** model and the **CODASYL** standard. These models were represented by the major databases of the day **IMS** and **IDMS**.

These systems were great advances on their predecessors but had significant drawbacks. Queries needed to be anticipated in advance of implementation, and only record-at-a-time processing was supported. Even the simplest report required programming resources to implement, and all IT departments suffered from a huge backlog of reporting requests.

### The relational model

In 1970, Edgar Codd wrote his seminal paper "A Relational Model of Data for Large Shared Data Banks”[[2]](#footnote-2). This paper outlined what Codd saw as fundamental issues in the design of existing DBMS systems:

* Existing DBMS systems merged physical and logical representations of data in a way that often complicated requests for data and created difficulties in satisfying requests that were not anticipated during the physical design.
* There was no formal standard for data representation. As a mathematician, Codd was well versed in formal data structures; he felt these structures had a role in Data management systems.
* Existing DBMS systems were too hard to use. To fulfill their potential, DBMS systems needed to be accessible to those without advanced programming skills.

The relational model described a means of logically representing data that was independent of the underlying storage mechanism. It required a *query language* that could be used to answer any question that could be satisfied by the data.

The relational model defines fundamental building blocks of a relational database:

* **Tuples** are a set of **attribute** values. In an actual database system, a tuple corresponds to a **row**, and an attribute to a column **value**.
* A **relation** is a collection of distinct tuples and corresponds to a **table** in relational database implementations.
* **Constraints** enforce consistency and define relationaships between tuples.
* Various **Operations** are defined such as joins, projections, unions, and so on. Operations on relationas always return relations. In practice, the output of a SQL query returns data in a table like structure.

Figure: Data represented in relational structure

Data represented in relational structure

### The SQL Language

Codd had specified that a relational system should support a “Database Sublanguage” to navigate and modify relational data. He proposed the Alpha language in 1971 which influenced the QUEL language designed by the creators of Ingres – an early relational database system developed at the University of California.

Meanwhile, researchers at IBM were developing **System R**, a prototype DBMS based on Codd’s relational model. They developed the SEQUEL language as the data sublanguage for the project. SEQUEL eventually was renamed SQL, and was adopted in commercial IBM databases including.

By the end of the 1970s, SQL had won out over QUEL as the relational query language became an ANSI standard language in 1986.

SQL needs very little introduction – today it’s one of the most widely used computer languages in the world. However, given the importance of SQL to CockroachDB, we’ll devote much of Chapter ?? to the CockroachDB SQL implementation.

### ACID transactions

The relational model and the SQL language represented two important foundations for the emerging databases of the early 1980s. The **ACID transaction model** represented the final piece of the puzzle.

All databases have to handle concurrent data change requests in a way that balances **consistency** with **concurrency**. In 1981 Jim Gray articulated the core principles of transaction processing that we still use today[[3]](#footnote-3). These principles later became known as ACID – atomic, consistent, isolated and durable – transaction processing.

As Gray put it “A transaction is a transformation of state which has the properties of atomicity (all or nothing), durability (effects survive failures) and consistency (a correct transformation).” The principle of Isolation required that one transaction should not be able to see the effects of other in-progress transactions.

### The RDBMS hegemony

The combination of the relational model, SQL language and ACID transactions became the dominant model for new database systems from the early 1980s through to the early 2000s. These systems became known generically as Relational DataBase Management Systems (RDBMS), though it’s important to realize that the relational model is just one of the three important pillars of almost all RDBMS systems.

By the end of the century the RDBMS reigned supreme. The leading databases of the day – Oracle, Sybase, SQL Server, Informix and DB2 all competed around performance, functionality or price, but all were virtually identical in their adoption of the relational model, SQL and ACID transactions.

It helped that the RDBMS came into prevalence at around the same time as another seismic paradigm shift. The world of Mainframe applications was giving way to the client server model. In the client server model, application code ran on microcomputes (PCs) while the database ran on a minicomputer, increasingly running the UNIX operating system. During the migration to client-server, mainframe based pre-relational database were largely abandoned in favor of the new breed of RDBMS.

### Enter the Internet

During the first decade of the 21st century an even more important shift in the application architecture occurred. That shift is of course, the internet. Initially, Internet applications ran using an stack not dissimilar to a traditional application. A single large sever hosted the applications database, application code ran on a “middle tier” application server such as WebSphere or WebLogic and clients interacted with the application through web browsers.

In the initial era of the internet, this architecture sufficed – though often barely adequately. The monolithic database servers were often a performance bottleneck, and although standby databases were routinely deployed, a database failure was one of the most common causes of application failure.

As the web grew, the limitations of the centralized RDBMS became untenable. The emerging social network and e-commerce sites had two characteristics that were increasingly difficult to support:

* These systems had global or near-global scale. Users in multiple continents would be required to simultaneously access the application
* Any level of downtime was undesirable. The old model of “weekend upgrades” no longer worked. There was no maintenance window that did not involve significant business disruption.

Internet pioneers such as Amazon

1. <https://commons.wikimedia.org/wiki/File:Cuneiform_tablet-_administrative_account_of_barley_distribution_with_cylinder_seal_impression_of_a_male_figure,_hunting_dogs,_and_boars_MET_DT847.jpg> [↑](#footnote-ref-1)
2. <http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf> [↑](#footnote-ref-2)
3. http://jimgray.azurewebsites.net/papers/thetransactionconcept.pdf [↑](#footnote-ref-3)