Cooperative Database Systems

Ideas for database systems that respect user data ownership.

By Randy Le

2019-05-13

Contents

Overview

This paper lists several ideas for what the author calls a "cooperative database system" (CDBS) – that is, a database system that inherently understands that the data it contains can have multiple owners.

The motivation behind this idea stems from recent revelations of abuses of private data from major internet companies. Facebook, Amazon, Google; all internet companies that have tons of data about their users, and users often have little say in how the data that is being collected by them may be used.

The author of this paper believes that the solution for this problem should be multi-pronged.

One, more user education and being more transparent in agreements between various technology companies and their users.

Secondly, the author believes that government oversight over the technology industry should also be available. Much like there is a Food and Drug Administration (FDA), a Federal Aviation Administration (FAA), there should be something akin to a "Federal Cyber Security Administration" to oversee the applications of technology in the industry. The scope as well as the pros and cons of such an agency are not discussed in this paper.

Finally, the other solution can be applied to how we engineer our database systems, the subject of which is discussed in this paper. Maybe database systems can be engineered so that users have greater control over their data.

Problem Statement

As the internet has continued to grow so has the amount of online data about its users. Often, various internet companies collect information about the users of their systems with little to no regard for the privacy and rights of their users.

In the past, database systems have been written primarily with a focus on organizing data and the speed in which data can be read, updated, added, and deleted from these systems.

This paper describes various concepts for a database system that instead of focusing on organizing and retrieving data instead focuses on data ownership – often the data that is stored in a database system may have multiple owners. This is with respect to the idea that someone hosting a database system does not inherently have all rights to all the data in the database system.

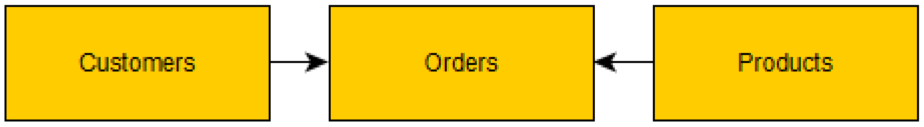
Acknowledging Design Flaws

The author of this paper acknowledges that despite the best intentions, there is no fail proof system. Once someone has been granted the ability to read a data point, even once, that there is little control one can have over how that data will be used. Nothing prevents a nefarious party that once has access to a system to simply copy off that data and archive it off for their own purposes.

The goal of a CDBS is to simply provide a potentially better way of organizing data that somewhat respects privacy of its users and grants rights back to users of a database system to do things with their data; regardless of the party that is hosting the database system. To do something such as "siphoning" data off in a CDBS would be discouraged by how simply inefficient it would become resource wise. A CDBS distributes data versus having private data in a centralized aggregate form.

Traditional Database Design (Relational Database Systems)

Consider a very simplified diagram of what a relational database system (RDBMS) may look like for an online retailer. (The diagram has been simplified from what may be considered an appropriate normalized form of a database to illustrate some main points.)



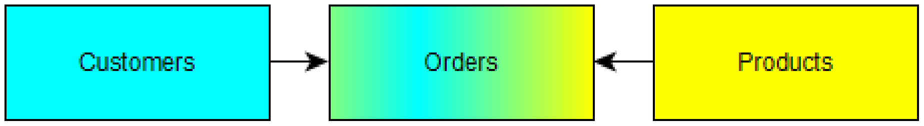
In the above example, there is a Customers table that the database has that contains all Customer information – perhaps their First Name, Last Name, and other items.

In another table, the database contains all the Products that the online retailer may be selling to its customers online.

Finally, there is an Orders table which contains all the orders that Customers have placed for various Products of the online retailer.

Data Ownership

Let's consider for a moment what may be argued about who "owns" what tables in this above database.



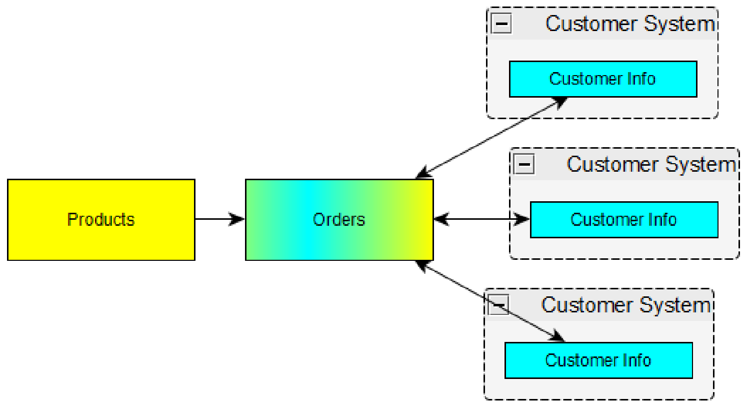
In the above colored diagram, the yellow Products table may be owned by the host of the database, in this case the online retailer. In the teal Customers table, the information about each customer may be argued should be owned by each individual customer. Finally, in the middle of the diagram there is a hybrid coloring which denotes that Orders are both shared by the host of the database – the online retailer, and partners of the host of the database system – the customers of the online retailer.

Defining Data Ownership

What does it mean when we state that a user "owns" their data? It should mean that the user can do CRUD operations on their data – create, record, update, and delete operations on their own data points. In addition, the users should have the ability to dictate who can access these data points. They should be able to grant access to their data as well as revoke access to their data.

A Database Design with Multiple Owners

Let's consider what a database system might look like if we could respect ownership of the Customers table for multiple owners (customers).



Consider the above diagram. This model remains mostly the same, except that now each Customer hosts their own local copy of a Customer table that the Orders table has references to. This is the main principle behind the idea of a "Cooperative Database System" (CDBS) – the data points that pertain to "partners" in a CDBS have full control over their data, but can still participate with a "host" of a database system; in this case, the online retailer.

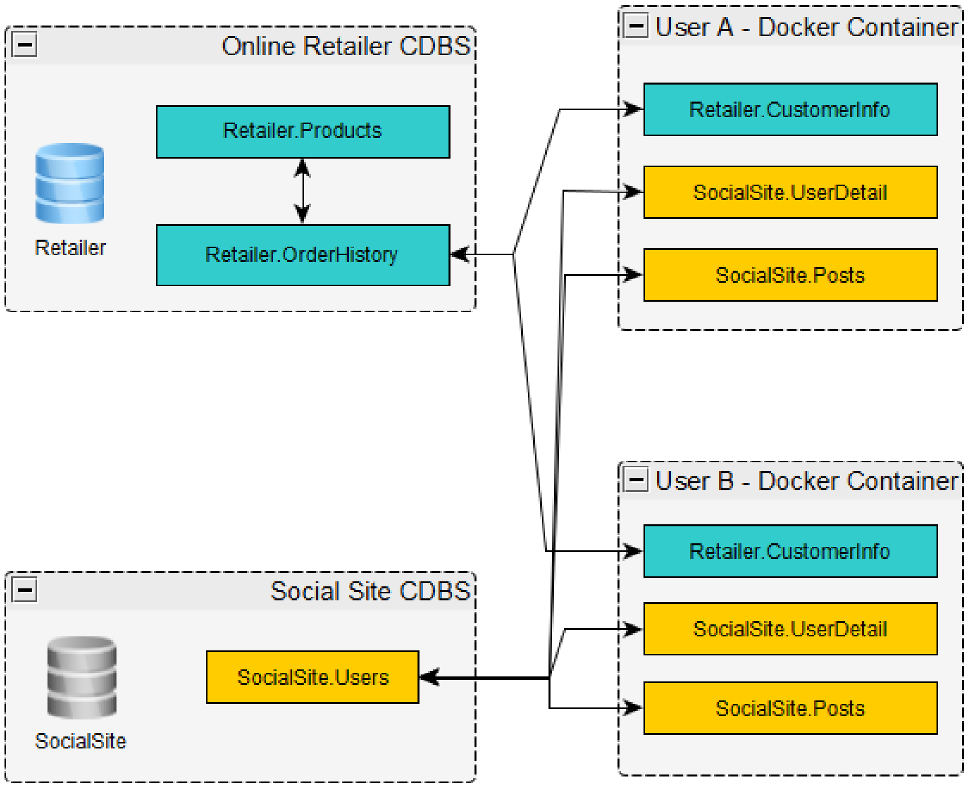
Potential Problems with Multiple Distributed Owners

Immediately, a few technical problems come to mind when considering the above model.

- Referential Integrity – if data is spread to many "partners" in a database system, can we guarantee data consistency throughout the entire database system? If data is spread out among many parties, how do we know that the data hasn't been manipulated in some way that is inconsistent with the rest of the system? Also, generally, how can we be sure the schema is correct throughout the entire database system?
- Performance – the ability diagram does not mention if the "partners" in this database system are hosting their own copies of their data points, or if the data lives within the host's machine that the users are only guaranteed rights over. If data is traversing over a network connection, the entire database system will not be very performant at all.

A Cooperative Database System

Expanding on the previous example, consider if a social media site and an online retailer site had their data systems constructed in the following manner –



In this example, an online retailer has in their hosted CDBS the tables for Products and Order History, but not any of the customer's personal information (First Name, Last Name, etc.)

In the other example, a social website has a table of users, but does not actually contain any information about the user's themselves nor does the database host have any of the user's actual posts in their system.

Basic Terminology

Let's define some terms for the above model. These terms will be used throughout the rest of the paper.

Host – a host is a party who both hosts most of the data and defines ***data contracts*** – these are agreements between the host and their partners (defined next) about the shape of the data, what the data points are and where they will be stored. A host of a CDBS system may be an online retailer such as Amazon.com or a social website such as Facebook.com

Partners – these are typically the users of a cooperative database system. They exert control over their own data points and are free to modify their own data points. Partners in a CDBS will be the users of a website such as Amazon.com or Facebook.com

Cooperative Database System Features

Data Contracts

A data contract explains the agreement between a host and a partner the *shape of the data* (the schema) that will need to be retained by the partner as well as *what data points* (e.g. First Name, Last

Name, Phone Number, etc.) will be required to be stored on the partner's copy of the CDBS versus what will be saved on the host's copy of the CDBS.

Access Control Lists

Implicit in a data contract is the understanding of who has access to data that a partner has. An access control list is defined for a partners CDBS that defines who has access to the partners copy, for how long, and what they can do with that data (e.g. read versus update, etc.)

Tokenization

A host CDBS needs to have some ability to reason over the data points that are stored on its partners copy of the CDBS. In order to respect its partners privacy, tokens may be used as placeholders for the actual content on the host's CDBS. An example of a token may be that the host CDBS may have a "token" table for all its partners of Customer information:

Id	First Name	Last Name	Age
1	Asdf	Aaa	1
2	Xyz	Bbb	3
3	Qwerty	Mmm	999

Caption: A "tokenized" table

In the above example "Customers" table above, the values are not the actual values – they are tokens or placeholders that should correspond to an actual value that is persisted on a partner's CDBS. Tokens are useful for the host to verify that it has valid data, without the need to read the value from the partner unless absolutely needed. It also ensures that the data types agree between the host and its partners.

Partner Id	"Token" First Name	Actual First Name	"Token" Age	Actual Age
1	Asdf	Brian	1	29
2	Xyz	Randy	3	33
3	Qwerty	Joseph	999	43

Caption: Each partner of a CDBS knows the value of the token that was saved at the host which corresponds to the actual value of the partner.

Tokens need not always need to be 1:1 – that is the same token is re-used multiple times. A partner in a CDBS may decide to use multiple tokens at a host that map to the same actual values. The only goal for a token is that it needs to be valid – a token that was used at a host must always map to some value at the partner.

Publicly Immutable and Consistent Logs (A Public Distributed Ledger)

Suppose in a CDBS hosted by an online retailer that someone makes a purchase for a product, say a toothbrush, for some amount - say $5. The data contract for this transaction states that the data for orders will be duplicated and saved both at the host and at the partner. How can we ensure that the partner doesn't later modify their copy of the order detail to attempt to later defraud the retailer – or vice versa? In other words, how can both parties be sure of the facts of the transaction, without publicly revealing details of the transaction? How can we settle disputes about the transactions that happen in a CDBS?

Borrowing ideas from come crypto-currency technologies, artifacts of the transaction must be persisted on a public distributable ledger. This public ledger can be backed by a *blockchain* system or a *directed* *acyclic graph* – the implementation technology is outside the scope of this document. The essential idea is that the transaction log is immutable and verifiable to all parties that participated (or are interested) in the transaction. A hash of the transaction is persisted onto the public ledger, of which the hash may consist of various data points – the product that was purchased, the time of purchase, etc. By putting such items on a public ledger, it ensures to all who wish to verify the transaction that an action did in fact take place, and that it must have included specific details which would have generated that specific hash. The hash of the transaction itself is also persisted at the host and the partner for reference.

In the same way that we can have public immutable ledgers of the details of the transactions (the transaction log), it may be useful to have a public immutable ledger about who accessed items from the host and from the partner (the access logs.)

Address Tables

Implicit in a CDBS system will be address tables, which will exist for any partner tables so that the host's database engine can refer to where data points are stored at a partner. Storing primary and secondary address in relation to replication (next topic) will be needed.

Replication

In order to improve performance and ensure redundancy of the entire CDBS, replicating a partner's portion of a CDBS may be needed. The author does not feel that this is a strongly needed feature but may be implemented later for performance reasons.

A replicated copy of a partners CDBS may be considered a secondary location or failover location versus the primary stored copy of the partners data.

ACID-ity of a CDBS

We should note that while having a public ledger helps provide verification and some consistency of our CDBS, it does not guarantee full ACID (Atomic, Consistent, Durable, Isolated) like properties that are traditionally found in other database systems. Since details of transactions are hashed before being put onto a public ledger, there is no way to rebuild a host's or partner's instance of the CDBS from the public ledger due to corruption of part of the system – intentional or accidental.

Indeed, ensuring full ACID properties of a CDBS is still under study by the author. Encouraging discussion of these implementation details is part of the intent of this paper.

Open Problems

Partner Locations

In this paper the author has not provided full details about how or where partners in a CDBS would persist their information. This is intentional. Keeping partner data in a CDBS local on a computer owned by a partner (i.e. a local workstation in their home) would be one way to fully ensure a partner had almost full control over their data but would introduce terrible latency issues in the entire CDBS.

On the other hand, if we kept partner data on the host's computing systems (i.e. locally on a retailers computers, such as Amazon.com or Facebook.com), it may improve performance of the overall CDBS, but we start to lose the ability for partners to truly ensure that their data is in their control.

The author of this paper wishes to make this an open problem left to the participants of the CDBS by leveraging some virtualization technologies such as using Docker containers or Virtual Machines in the cloud. This would allow partner data to live closer to the host's systems while at the same time ensuring some control over their data files. In addition, by virtualizing the container where the partners data lives, it can be easily moved to fit the logistical situation.

Finding the appropriate level of abstraction and distance from a host's part in a CDBS is understood to be an open problem.

Encryption

The author believes that ideally the data that is persisted by partners would physically live in an encrypted format on disk or in memory. This presents potentially even more latency issues in the operation of the entire system. However, given that partner data may live on disks not directly under their control, this may be an acceptable tradeoff.

The author likewise does not specify an encryption method to be used. The end goal is that data on disk, regardless of where that disk lives, if copied off would not be in a readable format without proper authorization.

Transport

This paper does not make any recommendations on the transport layer implementation for a CDBS. Transporting data in a CDBS would involve details mainly around –

- **Encryption** – how can we ensure data in flight is not seen by parties not participating in transactions?
- **Authorization / Verification** – how can a CDBS be sure that the host or partner that they are talking to is who they say they are? In other words, how can we be sure we do not allow impostors in our CDBS?
- **Performance** – with as much latency as a CDBS introduces by the very nature of the distribution of data – how can we minimize the amount of time it takes to transmit data between a host and a partner?

The author of this paper does not decide if the answers to the above details require authoring a new type of communication protocol specific to the concept of a CDBS, or if these implementation details can use existing technologies (TCP, HTTPS, etc.) to properly handle the problems that the concept of a CDBS introduces.

Future Features

Common Data Model

If CDBS systems are utilized in the future, it will be inevitable that some data points needed by multiple hosts for their partners will be redundant – customer first name, customer last name, shipping address, etc. While not necessarily needed, the author believes it may be useful to make a CDBS also understand the concept of the Common Data Model, a format recently introduced by Microsoft and its partners.

This concept provides an open specification of often used entities that should be interchangeable between multiple retailers.

Closing Notes

Usage of CDBS

The author believes that if applications were built upon a CDBS style of database that it would enable users of applications (web, desktop, etc.) to be more acutely aware of their privacy needs.

Going back to the original idea of an online retailer that built a web front on top of a CDBS system, rather than running analysis of “Recommended for you” for purchases on a centralized database system, users could instead request that the online retailer provide an algorithm and a data set that could be downloaded to the partner’s end of a CDBS, so that partner’s privacy could be protected while still offering insights to partners/customers of products that they may want or need.

Additional Business Models

Data Management as a Service

Managing partners instance in a CDBS setup will require a lot of infrastructure and potentially a lot of maintenance for each partner. In addition, ensuring the minimal amount of latency means that partners will need to have their end of a CDBS system as close as possible to the locations of server of the hosts of the CDBS.

The author imagines a new kind of data management service offering as a business. Consider a service akin to how banking systems work. Banks provide multiple services related to storing and servicing capital for their customers. In the same way, a data management service could be offered to users in a CDBS style world where partners need not worry as much about keeping latency low to their hosts, or maintaining data contracts by hand, or keeping up with access control lists. Instead, a data management service would offer a one stop shop where partners of a CDBS system could go to figure out everything about their digital information online and maintain their control over how it is used.