Data Resource

Alex Sverdlov
alex@theparticle.com

1 Introduction

Data is perhaps the most important resource that must be actively managed and maintained by all corporations. Often it is as simple as recording all business level events—in an append-only table or file.

2 Data Model

Databases store data in some structure—and every business needs to define that structure that matches the business model as closely as possible. There are several ways of going about this: recording the state or record changes to that state.

2.1 Maintain State

The database could present a state of the world as it exists right now. When the world changes, the database is updated (with updates or deletes) to match the new state.

For example, if the database has a list of customers, and a customer changes their address, then our customer-management application simply does an update statement on the customer record to change their address. Anyone accessing the customer table will now get the new address.

This view presents several problems: what if we need to know the customer's previous address? We might also want to do analysts on how often customers move (perhaps customers who change addresses often are not a good credit risk?). The point is that losing information is not a good approach.

One solution to this is to maintain some history. Every record might have start time and end time. So to change the customer's address, we would update the old record to have an end time of now, and create a new record with start time of now, and no end time.

While this approach works, it does require updating an existing record (namely, the timestamp).

2.2 Maintain Events

The other alternative is to store business events, and calculate the state on the fly when needed. For example, if a customer address changes, we simply add a record indicating that at this time, the customer's address changed to this new value.

This database is write-only: there are no updates/deletes made to the data once it is written. This creates a lot of flexibility—as we never have to update old data—the old data can be replicated and is always 'current' (as it was at the time it was created).

3 Database Programming

Database programming is often accomplished with SQL statements—often chained one after another. For computations where SQL is too cumbersome, a procedural language such as Scala (running on Spark) may be more appropriate.

Applications that access the database should not be intermingled with the database core—application must be treated as disposable. The way the data is collected and published must be independent of how it is stored in the database—this enables the database to outlive most user interfaces of many applications.

4 Trends

The central database with attached disk arrays appear to be going away. Same for databases with NAS mounted data.

The trend is towards distributed databases, where the data layer and compute layer are relatively independent and can scale separately. This is the model of AWS (and most modern architectures). It is motivated by the fact that compute and networking speed scale at different rates from storage media.

The large capacity disks have increased 1000x in 20 years (10GB disk in 2000 vs 10TB disk in 2020), but writing/reading speeds have not seen much improvement: it is still around 100MB/second.

At the same time, network speeds have gone from Fast Ethernet (100Mbps) to 10Gbps speeds (100x faster), with data center network speeds being about 40Gbps and sometimes 100Gbps.

What this means is that data locality plays a lot less role than it did 20 (or even 15) years ago—since grabbing data off disk is much slower than pushing it across the network. This means that compute and storage really should be separated—as is done by AWS with EC2 (compute) and S3 (storage).