Letting the use case drive the database choice

A dbInsight research paper for Amazon Web Services

Executive Summary

Trigger

The changes to the global economy that are driving digital business are also driving demand for a new generation of scalable, cloud-ready, modern applications. Compared to traditional, internally-facing enterprise transaction applications, modern applications respond to a wide range of scenarios and data types that in turn drive varying requirements for performance, scale, and geographic reach. There is no single, silver bullet data management recipe for responding to use cases ranging from supporting millions of concurrent sessions for online gaming, to identifying patterns of fraud in financial transactions, and delivering consistent high performance for scenarios with unpredictable demand spikes. With a variety of options, ranging from traditional SQL relational databases to NoSQL key-value stores, document database, graph databases, and others, how can organizations make the right choices?

Our Take

The use case drives the choice of database. While there is no single formula dictating whether an enterprise adopts one or more databases to address their business use case or scenario, they should depart from preset notions and dissect the problem into its constituent parts, assessing the need for requirements ranging from ACID transaction consistency, referential integrity, latency, availability, and other factors. In some instances, a mix of database systems may prove the best solution, for example with web and mobile commerce applications that may require a mix of simple lookups, search capabilities, and in some cases, leaderboards for showing hot selling items. In other cases, a single solution may address the challenge where the problem is narrow and well-defined, such as sorting through the complex interrelationships of entertainment content.

The first step in choosing the data solution always starts with the business goal and then evaluating the tasks that are necessary to address it. In many cases, the tasks and the requirements for data may be varied. For example, in e-commerce, where the operations often involve a mix of stable and volatile data, there is need for simple lookups and search, and the ability to quickly sort through and retrieve frequently accessed data. With numerous options for databases available, enterprises need not make compromises on choosing the best data solution for the job.

Diverse use cases demand diverse responses

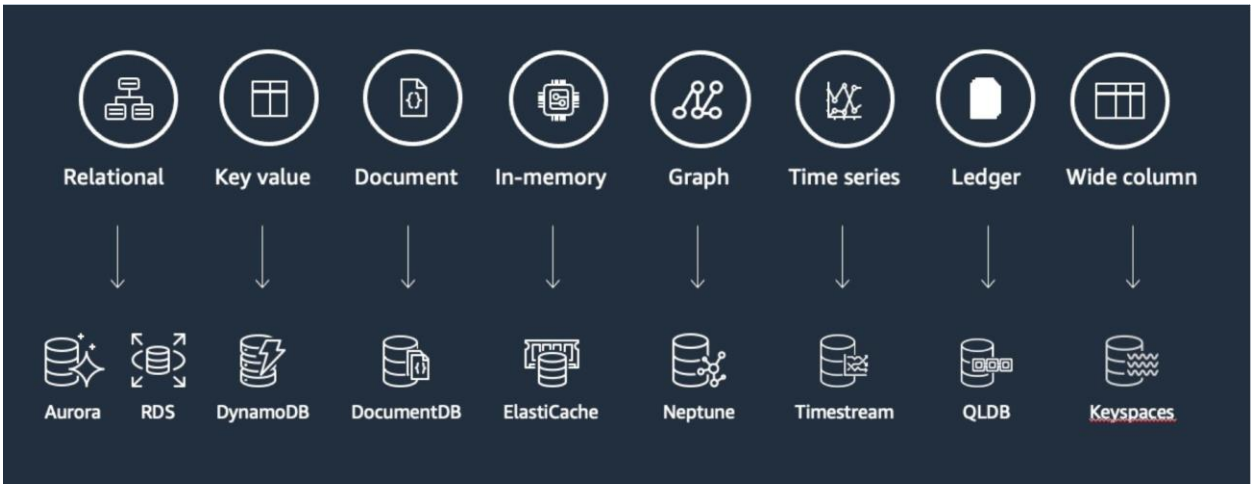
The requirements for modern applications

Gone are the days when a single database engine was the common answer for enterprise data management use cases. Relational databases remain the workhorses for many applications that require strict ACID guarantees and strict referential integrity ensuring that the references are valid, such as those processing financial transactions. They provide a *logical* means for accessing data, using a high-level declarative SQL language, allowing access to data without necessarily having to know where it physically resides.

But the rise of digital business, with the need for handling widely varying requirements for scale, performance, querying, data types, and more, has driven the need for alternatives to complement, not replace, the relational database. Today, organizations have more choices than ever – when it comes to selecting the database, they no longer need to fit square pegs into round holes. Enterprises should choose the right tool for the job. In some cases, they may need to select several databases because addressing the business problem may involve multiple tasks with divergent functions.

Figure 1 shows the portfolio of databases that are available from AWS to address different business problems.

Figure 1. AWS database portfolio



Source: Amazon Web Services

How digital business changes the ground rules

Compared to traditional enterprise transaction applications, modern cloud-based applications have a wide variety of performance and processing requirements. In its original retail business, Amazon learned that lesson firsthand, especially when dealing with extreme scenarios such as Amazon Prime Day. As a global event spanning over 400 fulfillment centers, there is the need to maintain consistent high performance regardless of demand spikes for performing product lookups, recommendations, and consummating financial transactions with referential integrity and ACID transaction guarantees. Because each part of the task for supporting Amazon Prime Day requires different capabilities, Amazon uses different databases for each part of the task, not to mention several other AWS services. For Amazon, that translates to 16.4 trillion database calls to Amazon DynamoDB across 66 hours (peaking at 180.1 million requests per second), while supporting fulfillment centers with over 1,900 Amazon Aurora instances.

For Duolingo, a mass provider of online language training, modern applications also require distinct tasks dictating a fit-for-purpose approach. Duolingo provides a choice of taking 98 different language courses, serving a base of 300 million students who, collectively, perform billions of exercises monthly. It must track student progress with an always-on operational database, using Amazon DynamoDB to keep students on the right path; Amazon ElastiCache for Redis to retrieve commonly-retrieved keywords to keep lessons rolling along; and Amazon Aurora PostgreSQL to provide the backend transaction processing system to run its global business. The challenge is similar for major travel booking sites that must quickly retrieve the search histories of individual customers, using a caching tier so that customer progress in booking rooms or plane fares will not be lost, complementing a transaction processing system for securing the reservations. The common thread for each of these cases is that, when running a highly scaled online business, each of the tasks that are performed are not readily handled by a single database with a singular data model.

Choosing the right path

The digital transformation occurs at different paces in different organizations. For some, the spark percolates up from lines of businesses where a team is responsible for kickstarting a new product or service, creating new mobile engagement with customers or business partners, or creating new marketing initiatives spanning social media and mobile in addition to traditional channels. In other cases, it comes from a major strategic pivot from the C-suite, where corporate strategies to embrace digital business originate.

Either way, the first step is to identify the business goal and the operational and technical challenges that may be standing in the way. Then the process turns to evaluating the existing portfolio of databases and applications, identifying costs and bottlenecks, then dissecting the

business problem or workflow into a series of tasks. For many organizations, it means departing from preset notions based on years of experience with relational databases as the default. Relational databases continue to have central roles, but for portions of the task that require always-on, scale, and extreme low latency, other options should be on the table. Having identified candidate solutions, enterprises need to understand the operational requirements, such as the likelihood for the need to add or drop compute nodes, distribute data, and tier storage, while keeping software updated and patched. In most cases, the clear answer is that these tasks will distract enterprises from their goal for embracing digital business. Managed cloud services free the organization from having to manage capacity, system configuration, load balancing, and software maintenance, enabling organizations to shift their focus from keeping the lights on to innovating their business. These include a broad range of managed database services purpose-built for different use cases.

To better understand how database choice plays out within modern organizations, let's take a look at some example design patterns within sample use cases.

Deep dive: Use cases

Web commerce

Web commerce applications, which on the surface have a single purpose for selling products online, have highly diverse functional requirements when you peer under the hood.

It starts with the need to connect customers with products. That requires a **product table** that, for some commerce providers, may amount to huge tables numbering tens of millions of SKUs. Each SKU has a unique identifier and, in most cases, multiple attributes such as product category, name, packaging, model number, and price. This lends itself well to a key-value database such as Amazon DynamoDB, because all queries are simple lookups; you locate a unique identifier that signifies an individual product or SKU and retrieve all the attributes associated with it. For instance, you choose a make and model of a specific road or trail bike; because the query is simple (just asking for the make and model), it can be retrieved more efficiently when compared to a relational database, where such a query might require joining a manufacturer table with a product model table to generate the result.

Traditionally, product tables were handled by relational databases, but when applications are exposed to extremely unpredictable traffic spikes, delivering consistent fast response requires a database that can denormalize data and scale out – exactly the attributes that a key-value store is well suited to provide. Additionally, if the key-value store is deployed as a serverless solution as with Amazon DynamoDB, enterprises can focus on defining service levels without

having to calculate the number of required nodes. If you add a mobile commerce channel to complement the website, there may also be the need for a caching service like Amazon ElastiCache that can further accelerate performance (e.g., millisecond latencies) to compensate for latencies introduced by the mobile network.

While some customers enter the website knowing which product they want, others often require a **product search** to find the best item for their needs. This involves a highly variable form of access, where query patterns cannot easily be predicted, making it a prime candidate for a search index. This search index can be designed to get real-time updates from the key-value database as the product table is updated, ensuring that search results will always be current. For instance, when looking for a bike, you might want to compare based on frame type (e.g., aluminum or carbon), frame size, brakes (hydraulic, rim, or disk), derailleurs (number of speeds), and wheelsets. When a product is updated, such as the introduction of a new disk brake, real-time updates from the key-value database will ensure the most current result.

When customers buy products, they often want to know which items are the most popular to help validate their choices. That is where **leaderboards** come in. Leaderboards require a high-performance engine for quickly accessing frequently-queried data, but without the overhead of scanning a full list of all products to find the champion. That is where a combination of two databases – the Redis open source, in-memory database that is updated in real time from Amazon DynamoDB – can address the challenge. Redis, which is available as an engine in the Amazon ElastiCache service, has a "sorted sets" feature making it optimized for storing scoring data; when paired with real-time updates from Amazon DynamoDB, leaderboards are constantly updated, and thanks to in-memory storage, instantly retrievable. The result is that scores are always current, a feature that is also useful for online gaming use cases.

Customers buying products often want **recommendations** that either guide them to the choice best suiting their needs, or to related products that are popular among customers with similar needs. Building a recommendations solution starts with a **customer profile** that is used as the basis for segmentation analytics that in turn can be used to shape recommendations. Document databases like Amazon DocumentDB (with MongoDB compatibility) have flexible JSON data models that are well-suited for customer profiles.

Activities such as product searches and purchases can readily update the customer record without having to change the data model. Analysis of the collected behavior can be fed to customer segmentation models that in turn can be used for generating product recommendations. Related use cases could analyze behavior, as collected in the customer profile, for potential churn analysis or, if the behavior hits an outlier, fraud detection. Graph

databases such as Amazon Neptune can also play a crucial role here; traversing a graph to identify other purchasers of the same item yields information on other products that they bought, and the results can then be ranked (often through another use of leaderboards).

Contact Tracing

With growing awareness and concern over public health, the role of contact tracing has risen to the forefront. For infectious diseases, contact tracing can provide a roadmap for the spread of illness and provide guidance on the level of risk. The pattern and nature of connections between people can be used to understand the level of risk and whether the risk level is increasing or declining, which in turn can be used for predicting future course of the disease and shaping public policy measures. The data may be collected manually or, in jurisdictions that permit it, through automated data collection from mobile devices. Contact tracing is intended to provide answers for questions such as:

- Where people are located and what is their status, based on contact history;
- Geographic risk levels based on presence of disease and protective measures taken;
- The degree to which people are complying with local restrictions.

This is a classic problem for graph databases like Amazon Neptune, as it traces the many-to-many relationships that are difficult, if not impossible to manage using conventional relational databases. When tracing interpersonal contacts, variables such as time, intensity of contact, and geolocation are relevant factors; each of these attributes can be a node or property that can be queried and tracked. And from that, the contacts of those whom the person has intersected by intensity (e.g., the length of time they spent with the contact) and/or location can be tracked in a similar manner.

In conventional relational databases, this would require the intricate joining of dozens, hundreds, or thousands of tables, whereas in a graph database, queries are matters of making a few "hops" to adjacent contacts based on attribute, such as length of exposure and/or geolocation. In so doing, the scope of infectious exposure can be determined, and hopefully, people who have had close contact with the infected can be advised on which measures to take.

Case studies

Caresyntax: Finding the right mix of operational and analytic data stores

Choosing data solutions requires understanding the data *and* how it will be consumed. Caresyntax delivers solutions for healthcare providers for optimizing their operating rooms. For them, the scalability of the cloud not only impacts the capacity to handle demand spikes, but also the ability to deliver far more granular analysis of the factors that impact both the

operational efficiency and the outcomes in its business. The cloud's scale offers another key benefit: elimination of process bottlenecks for data systems based on legacy on-premises architectures.

In 2019, Caresyntax completed a major acquisition that filled a key gap in its business: adding operational analytics for tracking the efficiency by which hospitals book their operating rooms. That complemented its existing business that had been focused on tracking and providing analysis of clinical outcomes. For most hospitals, surgery accounts for the lion's share of billing, equipment, and utilization of specialists. Successful hospitals efficiently schedule and utilize their operating theaters and consistently deliver successful patient outcomes. While analysis of operational efficiency and clinical outcomes have traditionally been highly siloed, there are huge potential advantages if hospitals can understand the impacts of one on the other, and from that generate lessons and best practices.

Caresyntax's operational analytics solution for surgical facilities uses several AWS database services including Amazon Aurora MySQL, for transaction processing and handling light analytic queries; Amazon Redshift, for handling analytic queries demanding more scale and granularity than feasible on a transaction system; open source Redis for caching "hot" frequently queried data; and Amazon DynamoDB for managing the availability of reporting parameters.

Originally, the key to choosing the transaction database was driven by skills and scaling requirements. The team developed the implementation on a self-managed MySQL database, but as the service grew successful, it required a more scalable system to handle transaction and query traffic. Caresyntax's solution stores basic data on operating room utilization, and as such, fits the profile for a typical transaction database use case: high volumes of read/write/update/delete queries on relatively limited columns or fields. While the range of data is limited, the size of the transaction store is significant, numbering in the millions of rows.

With Aurora, AWS offers a service that provided full compatibility with the team's original MySQL implementation; the team could move to Amazon Aurora with MySQL compatibility without changing any lines of code or the underlying schema. They gained a database service designed for the cloud that leverages a smart storage tier for accelerating write commits, automatic three-way replication to speed up reads, and much higher availability than possible with the basic self-managed open source implementation. That is critical for a database with high transaction volumes and row counts. While Aurora is used for transactions, it is also used for lightweight queries such as capacity utilization of the operating room, which involves lookups of relatively few columns that are readily handled, even with transaction stores of up to millions of rows.

The data also drove the requirement for a separate analytics solution. The factors impacting operating room utilization venture beyond scheduling. It also requires tracking inventory along with the events involved with conducting surgery. That extends from the point where the patient is rolled into the administration on anesthesia, the conduct of the operation, and the steps required to transfer the patient to recovery. With so many data elements, analytic queries are more optimally handled in a columnar database that also has the capability to perform federated query to analyze occasionally-used "cooler" data. Federated query provides a more cost- and time-efficient alternative to constantly performing ETL and persisting cooler data in the data warehouse.

Caresyntax uses Amazon Redshift, a data warehouse service, for its ability to store massive amounts of data that includes far more entities, or columns compared to the transaction system. Besides operating room utilization, Redshift also is used for storing data encompassing the details of the equipment, implanted devices, and surgical steps or events. Consequently, while Aurora can be used for analyzing the level of overall utilization of the operating room, Redshift can drill down with far more granularity. For instance, it can help compare performance by individual surgeons, such as why the same (or different) surgeon took differing amounts of time to perform the same procedure, and in turn, analyze the sequence of steps to help formulate best practices. This level of detail would not have been possible with a traditional on-premises data warehouse that could not efficiently autoscale to manage the data and provide analytic access to it.

For now, Caresyntax uses Redshift strictly for analyzing data in place, but Redshift also provides future-proofing for a company that has data stored in Amazon S3 storage for its clinical care analytic solutions; in the future, Redshift Spectrum might be used to query metadata from video stored in S3 for Caresyntax's clinical analytics application, Qvident, that records video from surgery so that individual surgical steps can be analyzed.

Complementing Aurora and Redshift, Caresyntax also runs Redis in Amazon EC2 as a caching layer for "hot" data that is frequently queried. Additionally, Amazon DynamoDB is used for configuring reporting, an important capability that enables Caresyntax to support the varying reporting requirements across different healthcare systems – requirements that are continually evolving thanks to changing regulatory regimes. By using a NoSQL data store, Caresyntax can store requirements and not be burdened with the overhead of having to change the schema each time a report requirement changes.

Zulily: Online retailer keeps customers returning with targeted searches

Zulily, an online retailer selling merchandise to mothers and their children, draws customers by making shopping an event-based experience. As a members-only site, Zulily promises a personalized shopping experience, where each day brings new assortments of women's fashion, children's items, and home décor to browse. Zulily offers a "best price" promise to its

members, launching over 100 events a day and featuring 9,000 styles that are live on the site for typically 72 hours.

The online retailer has placed the bulk of its online infrastructure on AWS and uses multiple database services, including Amazon DocumentDB (with MongoDB compatibility) and Amazon Aurora MySQL; it also runs its own implementation of MongoDB on Amazon EC2 and uses a third-party data warehouse. Because the site is event-driven and refreshed every morning at 6am Pacific Time, Zulily relies on Amazon Kinesis Data Analytics to filter search events from clickstream data in Kinesis Data Streams. Kinesis Data Analytics helps Zulily easily write SQL-like statements to filter search events across huge data sets at scale. As customers log in to search or discover what's new, Zulily captures search keywords submitted by customers, looks up related brands and product categories, performs an inventory check and offers popular search keywords, brands, and product categories as search suggestions. Since it was recently introduced, Zulily's suggested searches feature has proven a big hit, with over 75% of Zulily's customers who use the search feature leveraging search suggestions as they shop.

The search suggestions provide a rich experience showing trending searches based on relevant keywords, brands, and categories. The suggested searches make liberal use of images and graphics that inspire customers. This new feature replaced a more monolithic search feature that was primarily focused on trending searches.

As is common practice in e-commerce today, Zulily performs extensive user research offline to understand the member's preferences, and that is the starting point for shaping the suggested search. The differentiation of suggested searches is based on how Zulily makes those preferences actionable.

Behind the scenes, the choice of database and streaming analytics allowed Zulily to build this heavily-used feature faster and thus reduce the time to market. The team chose DocumentDB for storing search events, and Kinesis Data Analytics for smart filtering real-time clickstreams to extract search events that populate the database.

Here's how suggested search works. Any action that Zulily members take on the website or mobile app is recorded as clickstream events in Amazon Kinesis Data Streams. When a Zulily member requests a search, Kinesis Data Analytics filters relevant events from clickstream analytics. The microservice then feeds the search event to another Kinesis stream, which in turn triggers a Lambda function performing a lookup for relevant brands and categories in real time. The results are stored as enriched events in DocumentDB. These results populate the suggested search, which is composed of keywords, relevant brands, and categories. Once verified, the search results are populated back to the user in a rich display that, as noted above, carries additional relevant suggestions to provide shopping inspiration to customers. DocumentDB hosts the enriched search events and inventory checks are performed based on

data stored in DocumentDB. The final results, after inventory check, are stored in Amazon ElastiCache for Redis for fast persistence and are accessible to customers via a REST API.

Both DocumentDB and Kinesis were chosen for several reasons. DocumentDB was selected, first, because as a MongoDB-compatible database, it was familiar to developers on the team and would not require much of a learning curve. Secondly, as a highly scalable AWS-managed service, it would allow developers to focus on developing and iterating the application rather than having to shoulder the burden of deploying, operating, and maintaining the database. Kinesis Analytics was chosen to simplify the process of filtering out search events from the huge data set containing hundreds of millions of user actions and pageviews.

It was all part of Zulily's strategy to decouple development of new features to make the site, in essence, future proofed. New features would be developed through APIs to avoid the need for constant re-architecting of the underlying infrastructure. More importantly, because the team utilized managed AWS services, rather than having to handle all deployment, maintenance, and operation of the underlying data infrastructure itself, Zulily was able to get the suggested search feature into production in 10 weeks, which was a fraction of the time that would have been required if they went the conventional route.

Having gone live in August 2020, the new DocumentDB implementation of the customer experience database has proven its ability to keep the experience fresh during a period where online commerce has exploded. As proof of its popularity, the search suggestion feature is used by 75% of Zulily's customers using search as they shop.

Streaming Media service accelerates content personalization with a knowledge graph and machine learning

A streaming media service that is currently scaling up its operations by adding multiple new channel offerings is seeking to up its game for licensing content and matching it to viewer preferences. In the hotly competed streaming media market, making the right recommendation to the right viewer at the right time is the key to drawing customers.

The challenge is that categorizing content involves navigating through complex webs of interrelationships. The streaming service initially tried to manage content through a multi-model database, but found that the solution could not scale given the complexity of the data.

For instance, the listing for a specific show comprises season, episode, and genre. In turn, licensing that content for consumption must handle a wide range of client devices. Then there is the need to factor viewer history, preferences, and content viewed by people with similar tastes, or as part of similar demographic groups. The streaming service required a solution that could readily interface with machine learning predictive models that could readily adapt and predict trends in consumption habits and model the customer journey. And, in a medium where content is updated daily, there is the need to provide a leaderboard capability that can show top trending shows.

When AWS unveiled Amazon Neptune, a fully managed graph database service, this streaming media service signed on as an early reference customer. Amazon Neptune is a reliable, highly available graph database, supporting both property graph model and Resource Definition Framework (RDF)-style graphs, and the respective Gremlin and SPARQL query languages that are associated with those graph frameworks. The graph database's scaling capabilities have enabled the streaming service to track viewing habits far more closely – they can track every second rather than limit samples to every 5 seconds. Additionally, the ability to easily scale the cluster has also helped them handle the spikes in loads that special events – such as sporting events – regularly deliver.

For the streaming service, the graph data model was better equipped to handle all the complex interrelationships regarding categorizing content compared to the previous multi-model database. Taking advantage of Amazon Neptune's property graph support, the streaming service could work with a highly flexible and extensible schema that allowed the data model to evolve to support new relationships and categories – such as interest in event-based programming.

The REST API is used for surfacing program recommendations to the viewer. Amazon ElastiCache is used for frequently queried "hot" data. That same API is utilized for triggering machine learning models for surfacing viewing recommendations. The result, according to the project leaders, is that by using a database capable of representing the complex interrelationships between content, viewing habits, and streaming clients, the streaming service can be more responsible in attracting and retaining viewership.

Takeaways

Depending on the use case, there may be widely diverging requirements for managing and processing data. The data in a product lookup table that is stable will have different requirements from the data representing public health exposure from interpersonal contacts that are constantly changing. But the decision is not purely a matter of architecture. The same body of data that might have a relational structure may require different modes of access depending on whether the data is stable, such as a fact table, or volatile, such as tracking the latest product, gaming, or media content choices by the consumer. Can the requirements for reading and writing data be broken up into separate steps, such as member lookup and next-best action or recommendation, and therefore be addressed by database solutions that are designed for the purpose? Furthermore, does the nature of the use case translate to a need for extreme autoscaling and/or the need to read and/or write data with processes that are localized or highly distributed? And what is more important to the business case: returning data that is absolutely consistent, or ensuring that the service be highly available?

When choosing the database(s), look at the business use case first. With a multitude of database options out there, the choice does not have to be whittled down to an either-or.

Author

Tony Baer, Principal, dbInsight

tony@dbinsight.io

Twitter @TonyBaer

About dbInsight

dbInsight LLC® provides an independent view on the database and analytics technology ecosystem. dbInsight publishes independent research, and from our research, distills insights to help data and analytics technology providers understand their competitive positioning and sharpen their message.

Tony Baer, the founder and principal of dbInsight, is a recognized industry expert on data-driven transformation. *Onalytica* named him as one of its Top 100 influencers for data and cloud in 2019 and 2020. *Analytics Insight* named him one of the 2019 Top 100 Artificial Intelligence and Big Data Influencers. His combined expertise in both legacy database technologies and emerging cloud and analytics technologies shapes how technology providers go to market in an industry undergoing significant transformation. His regular ZDnet *"Big on Data"* posts are read 25,000 – 30,000 times monthly.

dbInsight® is a registered trademark of dbInsight LLC.