the GORILLA GUIDE® to...

Choosing a Database as a Service

Key Features and What To Look for in a DBaaS

JOEY D'ANTONI

Couchbase

POWERED BY ActualTech MEDIA

Choosing a Database as a Service

By Joey D'Antoni

TABLE OF CONTENTS

ACTUALTECH MEDIA

6650 Rivers Ave Ste 105 #22489 | North Charleston, SC 29406-4829

www.actualtechmedia.com

Publisher's Acknowledgements

EDITORIAL DIRECTOR

Keith Ward

DIRECTOR OF CONTENT DELIVERY

Wendy Hernandez

CREATIVE DIRECTOR

Olivia Thomson

SENIOR DIRECTOR OF CONTENT

Katie Mohr

WITH SPECIAL CONTRIBUTIONS FROM COUCHBASE

Matthew Groves, Sr. Product Marketing Manager

Rick Jacobs, Technical Product Marketing Manager

ABOUT THE AUTHOR

Joey D'Antoni is a Senior Architect and Data Platform MVP with over a decade of experience working in both Fortune 500 and smaller firms. He is currently Principal Consultant for Denny Cherry and Associates. He is frequent speaker at major tech events, and blogger about most technology topics. He believes that no single platform is the answer to all technology problems. He holds a BS in Computer Information Systems from Louisiana Tech University and an MBA from North Carolina State University.

Introduction

Database-as-a-service (DBaaS) offerings were among the earliest cloud Platform-as-a-Service (PaaS) offerings. The overhead of installing and configuring database servers, as well as managing maintenance activities—such as backups, patching, and supporting highly available solutions—means that there's a lot of value to be gained from using DBaaS offerings. Modern DBaaS solutions also offer value-added features such as scaling and query optimization that can help you improve the overall performance of your applications.

In an era when the data available to organizations is growing geometrically in size, many software engineers choose databases that aren't well suited to their workloads. Some even try to write their own data store as part of their application stack. It's crucial to understand the requirements of today's data and what each type of data store has to offer to meet the requirements.

A database platform should organize your data into structures such as tables, and provide data processing such as filtering and ordering data within a table or result set. This database processing is typically facilitated by query language of the database.

Cost is always a key element of any cloud services—and frequently the database is one of the most expensive resources in a cloud deployment. The reason for the

database's outsized cost burden is that databases consume a lot of hardware resources. Even well-optimized databases can be heavy consumers of CPU, memory, and storage. Although the initial cost for a development database can be very low, costs can increase dramatically as your database accumulates data and users and you must scale up or out. It's important to quantify how much that performance will cost; for instance, how much it will take to onboard 100 new users to your application.

Key Features of a DBaaS

Although many DBaaS offerings on the market tend to offer several common features, no matter the core database engine:

- Rapid Deployment
- Flexible Deployment
- Rich interfaces
- Scalability
- Administrative Services and Managed Maintenance
- Security
- Observability
- High availability

RAPID DEPLOYMENT

The administrator shouldn't have to install and configure software or create virtual machines or containers. The service should provision the database and all the resources that are needed to support it, and the service should be able to scale with minimal downtime.

FLEXIBLE DEPLOYMENT

Your DBaaS offering should be deployable across clouds and locally in your own environment, and ideally support multi-cloud architecture. Having a local environment lowers overall costs and increases development velocity. If your DBaaS can be deployed via an API, you can define your infrastructure as code in automated deployment pipelines and avoid being locked into a single cloud provider. Having a database that can be deployed flexibly also allows you to run anywhere, on any cloud. Many organizations want to be able to deploy their application stacks anywhere: Kubernetes, bare-metal, or public cloud.

RICH INTERFACES

The most common query language across all types of databases is Structured Query Language (SQL), an ANSI standard that has been in broad use for more than 30 years. Although the popularity of SQL has made it nearly ubiquitous, a database should ideally support multi-model ways of accessing the same pool of data through programmatic access, direct API calls, and various data services. The database should come with a software development kit (SDK) for different

popular programming languages. Application developers need libraries to access the database from their application source code, no matter what language they're using to build their applications, or which model they are using to connect to the same pool of data.

HIGH AVAILABILITY

The service should be built in a highly available fashion to ensure that any hardware or transient software failures leave your database up and running. A valuable feature in some DBaaS offerings is disaster recovery. This feature lets you span the service to another region, or even cloud provider, in the event of a system failure in your primary regions.

SCALABILITY

Any well-designed DBaaS offering can scale, whether through increasing storage capacity or the number of CPU cores dedicated to the service. The inherent high availability of DBaaS means that these scaling operations can be completed with minimal downtime, which ideally would not be affected by the size of the databases being changed.

ADMINISTRATIVE SERVICES AND MANAGED MAINTENANCE

The DBaaS platform should provide key administrative services such as automated backups and patching. Both the OS and the database software itself should have patching that happens without user intervention and minimizes

downtime. Backups should also be provided by the service and allow for point-in-time recovery with minimal administrative effort.

SECURITY

In a long-ago world (the year 2000), a major breach was driven by a configuration flaw in a prominent database platform: It didn't require its system administrator (which had a fixed username) to have a password. Modern security threats are more dangerous, but cloud offerings are also quite sophisticated in repelling the attacks.

A few features are absolute requirements, such as encryption between the client and database. Some features are absolute requirements for a database running in a cloud platform, such as encryption between the client and database.

Although the initial cost for a development database can be very low, costs can increase dramatically as your database accumulates data and users and you must scale up or out.

In the early years of cloud databases, the immaturity of the underlying cloud platforms limited the security of the database. For example, Microsoft's first entry into cloud databases, SQL Azure, initially didn't support private networking or encryption at rest. Some services still require access from a specific IP address at the vendor's site. But the

most secure pattern is to place your database on a private network, which prevents most brute-force intrusions. The current incarnation of SQL Azure supports private networking, encryption, and much more, but it took a long evolution

The DBaaS platform should provide key administrative services such as automated backups and patching.

to get to that point.

More and more databases support multifactor authentication for login. This dramatically increases security by limiting the ability of attackers to use brute-force attacks.

OBSERVABILITY

Database software can be very opaque to application performance—so it is of critical importance that your database be instrumented, allowing you to observe performance throughout the system. Ideally, this database telemetry can be shared to other monitoring platforms so that you can analyze any bottlenecks wherever they are in the application stack.

How To Pick the Right DBaaS

Your database has always been one of the most important parts of your application architecture. With NoSQL databases proliferating, and the ease of deploying additional cloud

services, application teams have many choices for storing data in their applications.

Choosing the right database for your application used to be simple: Pick a commercial or open-source relational database and live with it. While relational databases still have their use cases, modern applications employ a much wider variety of databases, whether they be document databases, key-value stores, or other data models. A document database usually stores JSON documents, which give developers agility in designing data models to match their applications while allowing them to evolve over time in a manner that is much more flexible than a relational table structure. Document databases and other NoSQL data stores allow

The primary objective for any DBaaS offering is to meet the design goals of your application, but important secondary considerations include lower total cost of ownership (TCO) and the pricing model.

changes in data structure to be made much more easily than with adding and removing columns from tables.

The primary objective for any DBaaS offering is to meet the design goals of your application, but important secondary considerations include lower total cost of ownership (TCO) and the pricing model. For example, you want costs to scale in a linear fashion as your users and utilization increase—not to shoot up exponentially. Typically, this means you want a DBaaS model that has a predictable pricing model:

You pay the same price for a specific set of resources no matter how much you use them.

THE IMPORTANCE OF ACID

One of the reasons for the ubiquity of relational databases in many legacy applications is because they have ACID properties. ACID stands for Atomicity, Consistency, Isolation, and Durability, which are properties databases need to meet to ensure, among other things, the protection and accuracy of transactions (see **Figure 1**). For instance, any application that supports financial use cases often needs ACID transactions. Although many early NoSQL databases relinquished ACID in favor of eventual consistency models in an effort to improve overall system throughput, most modern NoSQL offerings support adjustable consistency models and the ability to opt-in to ACID transactions.

Although deploying new database services may not be a normal part of your daily build process for your application, it's still important for your DBaaS to provide a quick deployment path, and support various deployment tools like Terraform let you deploy your infrastructure as code.

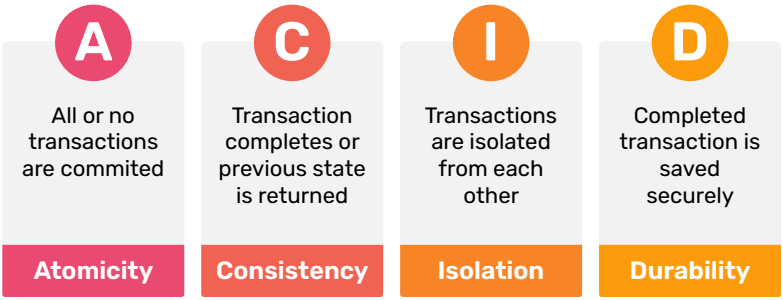


Figure 1: ACID properties in databases

Scaling operations should take place in minutes, and not have a significant impact on workloads. It shouldn't take six hours to deploy a new instance of your data service.

Another common use case for modern applications is to de-

ACID stands for Atomicity, Consistency, Isolation, and Durability, which are properties databases need to meet to ensure, among other things, the protection and accuracy of transactions

liver and synchronize data with mobile devices. Some mobile apps rely on having a database embedded on the mobile or Internet of Things (IoT) device that could be offline.

Support for multiple data access methods and linked services allows your database development to be more flexible and heavily reduces the complexity of separate dedicated technologies, such as dedicated full text search or caching engines like Redis.

PERFORMANCE AND SCALE

As mentioned earlier, databases can be heavy consumers of resources in your application. Even when database queries are well optimized, databases can run into performance issues as workloads increase. Troubleshooting database query performance frequently requires special expertise around query optimization and ensuring that correct indexes are

created for common query patterns. If your database isn't optimized, even though some users continue to experience good performance, it degrades as more users begin issuing queries.

Although most databases provide local high availability, not all are resilient in the face of disaster, where an entire data center may fail.

Your database should support robust SQL language options such as windowing functions, common table expressions (CTEs), and user-defined functions, which allow both developers and end users to develop robust queries quickly and easily.

In global applications, it can be very helpful to put a database closer to customers in specific regions. This geographic reach is one of the key advantages of the public cloud. Azure, Amazon Web Services (AWS) and Google Cloud offer coverage in all the major world regions. If you replicate your database in many geographic regions, customers connecting to application servers and content delivery networks in those regions experience lower response times than if those application calls had to travel back to a database in a central region that might add nearly a second of latency to the transaction. Additionally, privacy and security regulations

in some countries require user data to be stored locally within that region.

HIGH AVAILABILITY AND DATA RECOVERY

While most DBaaS offerings take care of high availability and backups, you should understand your vendor's service-level agreement (SLA) for those services. What happens when a node fails? What impact does it have on the overall service? What is the timeline to restore data if a deployment breaks existing documents?

Although most databases provide local high availability, not all are resilient in the face of disaster, where an entire data center may fail. Furthermore, disaster recovery services often must be manually configured and this increases the total cost of ownership.

COST MANAGEMENT

Because everything in cloud computing has a price tag, pricing and TCO are front and center of many decisions. Costs are especially important for a relatively expensive resource such as a database. So IT teams put a lot of focus on performance and the ability to support more users with fewer resources. In most cases this means more optimal database design, support for autoscaling as workloads grow, but the ability to quickly scale back as load decreases, or tunneling activities increase. The other cost aspect that you need to consider is storage cost: Can you scale storage in your database without having to scale other resources (for example, having to add 8 CPU cores to add an additional 2TB

of storage to your database) that your application might not require? You should ask questions around the anticipated costs of scaling your workloads to avoid these traps.

Common Use Cases and Recommendations

DBaaS offerings can speed up application development, so they're popular for many modern applications. Workloads vary, but the following examples show common areas where a modern DBaaS offering can have unique benefits:

- **Dynamic product catalogs:** These are a particularly good fit for document databases, which allow flexible schemas across different products rather than a rigid table structure.
- **Management of flexible user profiles and personalized digital experiences:** Like catalogs, user profiles have different sets of attributes depending on user locale or preferences, so a flexible schema simplifies development.
- **Mobile and IoT applications:** These applications have tremendous write scale that can challenge the performance limits of a relational database. Also, their distributed and sometimes online workloads means that they can sometimes benefit from embedding portions of the database locally.

- **Operational analytics:** By storing this analytic data in your DBaaS, you can customize both applications and user experiences more quickly by making data-driven decisions in real time, within your application code.

While these four use cases are some of the more common ones for DBaaS offerings, any application that writes data to a database can be a great fit. Reducing your administrative overhead and increasing the availability of your applications will improve the productivity of your development teams while at the same time help you meet your service-level objectives.

LEARN MORE

To learn more about Couchbase Capella:

- Sign up for a free 30-day trial if you haven't already.
- Take a look at the Capella Learning Path!
- Connect your trial cluster to the Couchbase Playground or connect a project to test it out for yourself.
- If you're already using Couchbase Capella, interact with your cluster by using the interactive Couchbase Shell or the Capella Control plane to:
    - Check out the document viewer
    - Connect to a project

About Couchbase

Unlike other NoSQL databases, Couchbase provides an enterprise-class, multicloud to edge database that offers the robust capabilities required for business-critical applications on a highly scalable and available platform. Couchbase is built on open standards, combining the best of NoSQL with the power and familiarity of SQL, to simplify the transition from mainframe and relational databases.

Couchbase has become pervasive in our everyday lives; our customers include industry leaders Amadeus, American Express, Carrefour, Cisco, Comcast/Sky, Disney, eBay, LinkedIn, Marriott, Tesco, Tommy Hilfiger, United, Verizon, as well as hundreds of other household names. For more information, visit www.couchbase.com.

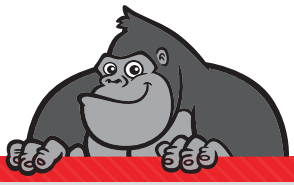
About ActualTech Media

ActualTech MEDIA

ActualTech Media is a B2B tech marketing company that connects enterprise IT vendors with IT buyers through innovative lead generation programs and compelling custom content services.

ActualTech Media's team speaks to the enterprise IT audience because we've been the enterprise IT audience.

Our leadership team is stacked with former CIOs, IT managers, architects, subject matter experts and marketing professionals that help our clients spend less time explaining what their technology does and more time creating strategies that drive results.



If you're an IT marketer and you'd like your own custom Gorilla Guide® title for your company, please visit https://www.gorilla.guide/custom-solutions/