David Chappell

OPERATIONAL SCENARIOS USING THE MICROSOFT DATA PLATFORM

A GUIDE FOR IT LEADERS

Sponsored by Microsoft Corporation

Contents

Microsoft's Data Platform: The Big Picture

Data is central to what we do. Even though we now call our profession "information technology" rather than "data processing", data is no less important. In fact, the volume, variety, and velocity of data that we work with increase every day. To deal with this, organizations use a range of data technologies in many different scenarios. Taken as a group, these technologies make up a *data platform*.

A useful way to think about the technologies in a data platform is to divide them into three categories based on the kind of data they work with. Those categories are:

- *Operational data*, such as transactional data used by a banking system, an online retailer, or an ERP application. This data is typically both read and written by applications, commonly in response to user requests. A banking application might read your account balance, for instance, then write a new value to reflect a deposit you make. And while operational data was once almost entirely relational, the increasing volume and variety of data have changed this. Today, working with unstructured operational data can be just as important.

- *Analytical data*, such as the information kept in a data warehouse. This data is typically read-only, and it usually includes historical information extracted over time from other data sources, such as operational databases. Analytical data is commonly used for things such as business intelligence and machine learning, and like operational data, it can be either relational or unstructured.

- *Streaming data*, such as data produced by sensors. The defining characteristic of streaming data is velocity; if the data isn't processed quickly, it loses a large part of its value. Many streaming scenarios today relate to the Internet of Things (IoT), where the focus is on interacting with data provided by lots of devices. Streaming data is also used in other situations, such as analyzing financial transactions as they happen. In both cases, the challenge is to work effectively with large amounts of data being produced in real time.

The Microsoft data platform provides technologies for all three categories, along with connections among the three. Figure 1 summarizes the platform's offerings in each area.

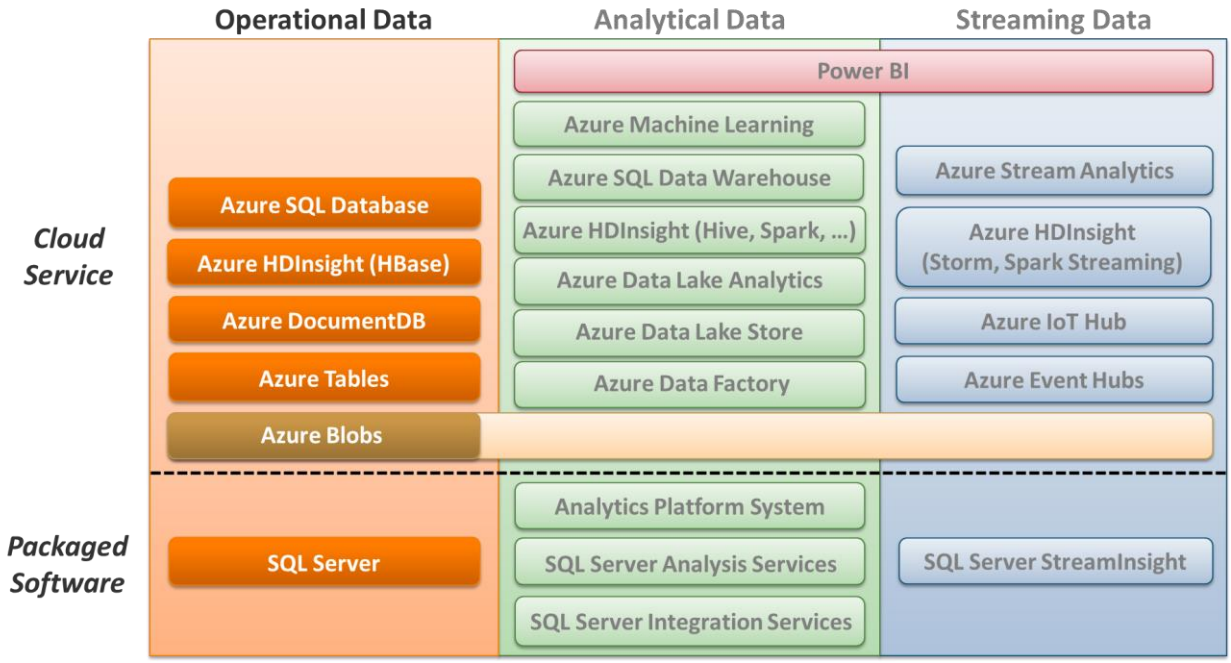

Figure 1: The Microsoft data platform includes cloud services and packaged software for working with operational data, analytical data, and streaming data.

This paper focuses on the leftmost column in the figure, Microsoft's offerings for working with operational data. (For more on the other two categories, see the companion papers *Analytical Data Scenarios Using the Microsoft Data Platform* and *Streaming Data Scenarios Using the Microsoft Data Platform*.) And don't be confused by the diagram: These technologies aren't layered in the sense that each one depends on the others shown below it. It's more accurate to think of each column as a group of technologies for working with data in a particular way. Also, realize that the lines between the columns are permeable—these technologies can be used in various combinations. For example, the operational technologies in the left column are often used together with the analytical technologies in the center column.

The easiest way to understand these technologies—and more important, to understand how they can help your organization—is to walk through scenarios that use them. And given how important cloud computing is to IT leaders today, those scenarios should all use the cloud in some way. Accordingly, this paper looks at four common operational data challenges faced by organizations today, describing how the Microsoft data platform addresses each one. Those challenges are the following:

- An enterprise moving an on-premises packaged application to the cloud.
- An enterprise creating a new internally facing application.
- An enterprise or independent software vendor (ISV) creating a customer-facing application.
- An ISV creating a new SaaS application.

Along the way, we'll take a brief look at each of Microsoft's operational data technologies. The goal is to provide a big-picture view of how the Microsoft data platform addresses the challenges of operational data.

Scenario: Moving an On-Premises Packaged Application to the Cloud

The debate is over: cloud computing owns the future. One way for enterprise IT organizations to take their first steps into cloud platforms is by creating a development and test environment using cloud VMs. Doing this can provide a faster and less expensive environment for building new software, as well as offering a low-risk way to get your feet wet with a cloud platform such as Microsoft Azure.

Another option is to move an existing packaged application and its data to Azure. Organizations do this to save money, to free up on-premises computing resources, and for other reasons. For applications built on the Microsoft data platform, that data is typically stored in SQL Server.

Technology Snapshot: SQL Server

SQL Server has been Microsoft's flagship database management system since its introduction more than two decades ago. During that time, the way its customers use it has changed dramatically. From its birth as a departmental server, SQL Server has grown to become an enterprise-scale solution, able to support very large applications. The product is focused on relational data, and so it provides SQL queries, stored procedures, secondary indexes, and more.

SQL Server was created in the pre-cloud era, but it also runs well on Azure. And despite Microsoft's full-throated embrace of cloud computing, the company clearly intends to keep investing in this database technology. You needn't worry, for example, about ongoing support for your SQL Server data, either on-premises or on Azure. Microsoft continues to add new features to this fundamental part of its data platform.

Describing the Scenario

If you decide to move an existing on-premises application and its database to Azure, you'd like to do it as easily and cheaply as possible. The typical approach for doing this is to use Azure Virtual Machines, a cloud technology that provides *Infrastructure as a Service (IaaS)*. IaaS creates VMs on demand, letting you request a new VM via a website, then have it available in an Azure datacenter in just a few minutes. Figure 2 illustrates a packaged application running on Azure in IaaS VMs.

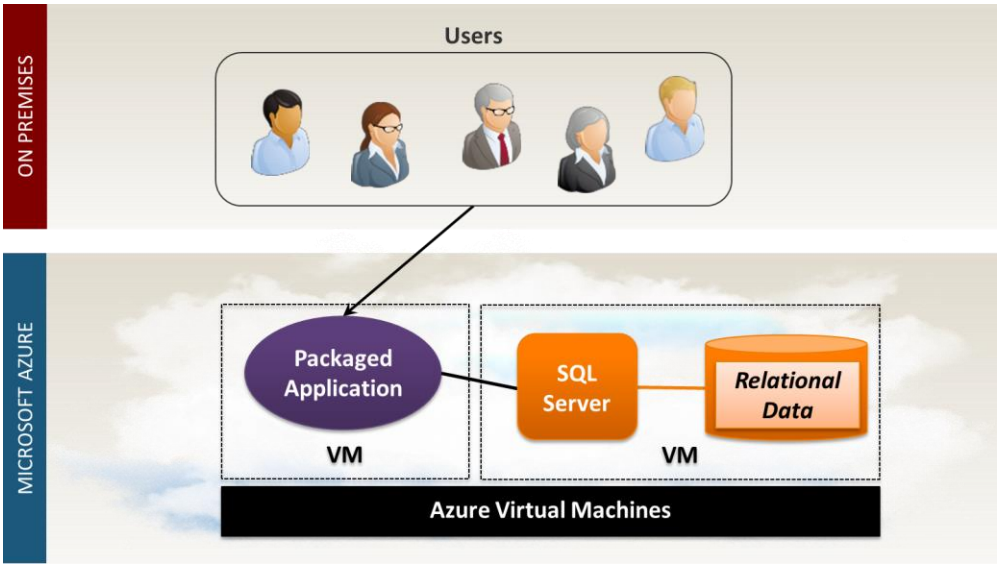

Figure 2: A packaged application moved to Azure can rely on SQL Server running in an Azure VM.

As the figure shows, the application might run in one VM, while SQL Server runs in another. Users of the application shouldn't notice that it's now running on Azure, and for the most part, neither will SQL Server. This database system runs well in on-premises VMs, and so running it in Azure IaaS VMs isn't a stretch. Azure VMs are based on Hyper-V, and Azure's physical machines run ordinary Windows Server, so running SQL Server in the cloud is much the same as running it on premises.

Understanding Your Options

Organizations move packaged applications and SQL Server to the cloud for a number of reasons. Some of the most common are the following:

- It can save money. Running on Azure isn't always cheaper than running in your own datacenter, especially if you've already paid for your servers. When it's time for a server refresh, however, it might well be less expensive to use Azure when you take all of the on-premises costs into account. And over time, competition among Microsoft and other large cloud platform providers will almost certainly make Azure less expensive than your own datacenter. You'll still need IT people to manage your Azure VMs, but expect the cost of running those VMs to go down.

- Moving a packaged application to Azure can provide a low-risk way to get started with the cloud. Most IT leaders agree that cloud computing is the future of our industry. Most also agree, though, that moving everything to the cloud immediately is too risky. Choosing one existing application to move is a good way to get started with this transition.

- Over time, moving on-premises applications to Azure will let your organization focus on its business rather than IT. Just as you buy electricity from a public utility instead of running your own generator, letting a global-scale datacenter specialist run your applications also makes sense. While the transition will take time (and some applications might never move), the idea is certainly attractive, especially to business leaders.

Scenario: Creating an Internally Facing Application

In a typical enterprise, a majority of the applications are internally facing—they're used by employees rather than customers. And of these internally facing applications, many are custom, built explicitly for this organization. Going forward, more and more people are choosing to build these applications on a cloud platform such as Azure.

In fact, when enterprises start building new software in the cloud, they often begin with internally facing applications. Starting here can be less risky, since a failure might not be as visible to customers. And while it's possible to build a new Azure application using SQL Server running in an IaaS VM, it might make more sense to use SQL Database instead.

Technology Snapshot: Azure SQL Database

SQL Database is a relational database offering provided by Microsoft Azure. To an application developer, it looks much like SQL Server, with full support for SQL queries, stored procedures, and secondary indexes. Unlike SQL Server, however, SQL Database is a managed cloud service. Rather than installing and managing a database server on a physical or virtual machine, you can just ask SQL Database to create a new database. That database is ready in a few minutes, with no installation required. SQL Database also handles much of the management work for you, making it faster and simpler to create and use relational databases.

Technology Snapshot: Azure Blobs

Relational services like SQL Database are the right choice for working with many kinds of data. Sometimes, though, a relational database can be overkill. For example, suppose an application needs to store simple unstructured data such as images or videos. The power and complexity of a relational database aren't necessary—a simpler and less expensive solution is better.

Azure Blobs can provide this solution. The term "blob" is an acronym for Binary Large OBject, and that's exactly the kind of data Azure Blobs works with. This storage technology stores raw binary information, and it can be useful in many different situations. It's also quite scalable and relatively inexpensive at just a few cents per gigabyte per month.

Describing the Scenario

Suppose your organization decides to create a new internally facing application on Azure. Suppose also that this application works with both relational data and binary data. Figure 3 shows how it might look.

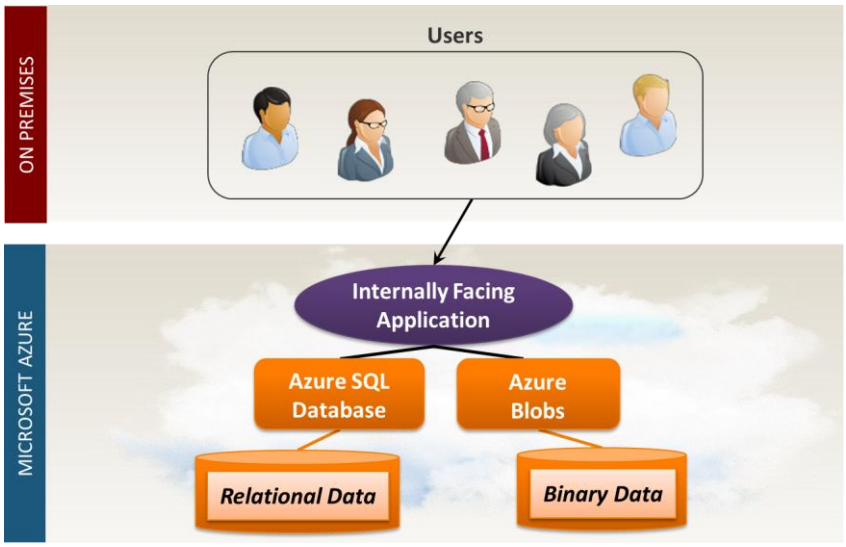

Figure 3: A new internally facing application might use both SQL Database and Blobs.

One possible example is an application that provides online access to recorded sessions from internal company conferences. The application might use SQL Database to store structured information about each session, things such as the session's name, its description, and who the speakers were. It might also use Blobs to store videos of each session, with a reference to the right blob kept in each session's SQL Database entry. As this example shows, combining different parts of the Microsoft data platform in a single application often makes sense, since different technologies work best with different kinds of data.

Understanding Your Options

Your organization might choose to build a new internally facing application on Azure to lower costs, to get faster access to computing resources, or even to get around barriers created by your central IT organization. Whatever the reason, the operational data challenge is to choose the right database for this new cloud application.

When should you build a new application using SQL Database rather than SQL Server running in a VM? In the majority of cases, the answer won't depend on technology. SQL Database implements almost everything that's in SQL Server, and Microsoft's goal is to make the two feature equivalent. Instead, the main drivers for the decision should be these:

- Control vs. administrative effort. With SQL Server, you'll have all of the control you're used to, but at the cost of significantly more administrative work. You must manage both the database server itself and, to some degree, the IaaS VM it runs in. SQL Database, by contrast, does much of this administrative work for you, including backups, making your life simpler and cheaper. And because it's a managed service, there are no VMs for you to maintain yourself.

- Need for fast access to advanced features. With SQL Server, providing high availability takes effort, expertise, and time. SQL Database, by contrast, gives you high availability, scale-out, and more automatically. This lets you spend your time and effort on things that provide direct business value rather than setting up infrastructure.

There are a couple of cases in which using SQL Database isn't recommended. If a new application needs specific database features that are only available in SQL Server, for example, using SQL Database isn't a great idea. Also, the maximum size for a single database today in SQL Database is one terabyte. If you need a database larger than this, SQL Server in a VM is a better choice. Still, SQL Database does provide an option for *sharding* a database, i.e., breaking it into multiple pieces, that allows much bigger databases.

For most internally facing applications built on Azure, it's fair to say that SQL Database is a better choice than SQL Server in a VM. It's quicker to get going, it's easier and cheaper to manage, and depending on how your application uses data, it might cost you less.

Scenario: Creating a Customer-Facing Application

For many companies today, customer-facing applications are the heart of their business. In the age of the internet, letting customers interact directly with the services you provide is often a good idea. Even for firms that don't depend on them entirely, customer-facing applications have become increasingly important.

Building a new customer-facing application on a cloud platform can be a good idea for several reasons. Cloud platforms like Azure scale very well, for example, and they provide a wide range of services. Once you've made the choice to do this, however, you face another decision: Which cloud technologies should you use for your application's data?

One option is SQL Database. It's scalable, straightforward to use, and requires minimal maintenance. Yet there are situations in which a relational approach isn't the best option for a new application. In cases like these, an organization might opt to use a NoSQL store instead.

Technology Snapshot: Azure NoSQL Stores

Microsoft Azure provides several NoSQL technologies for working with operational data. All of them are managed services, so there's no need to install or maintain a database server. The NoSQL options on Azure today are the following:

- DocumentDB, a document-oriented service that stores data as collections of JavaScript Object Notation (JSON) documents. It provides a SQL-based query language, support for stored procedures and secondary indexes, and the ability to create a database that holds hundreds of terabytes of data.
- Tables, a key/value store that stores data as entities accessed via unique keys. This service is in some ways simpler than DocumentDB—there's no support for SQL-based queries or stored procedures or secondary indexes. It does allow scaling a database up to hundreds of terabytes, however, and it's both very fast and quite inexpensive.
- HDInsight HBase, a column family store that's part of the Hadoop technology family. HBase excels at handling very large tables—a single table can have millions of columns and billions of rows—with each table holding hundreds of terabytes. It can support a subset of SQL queries, but HBase lacks secondary indexes and all but the most basic support for transactions.

Describing the Scenario

The Microsoft data platform provides both SQL Database and NoSQL technologies[1]. As Figure 4 suggests, a customer-facing application can use one or more of these.

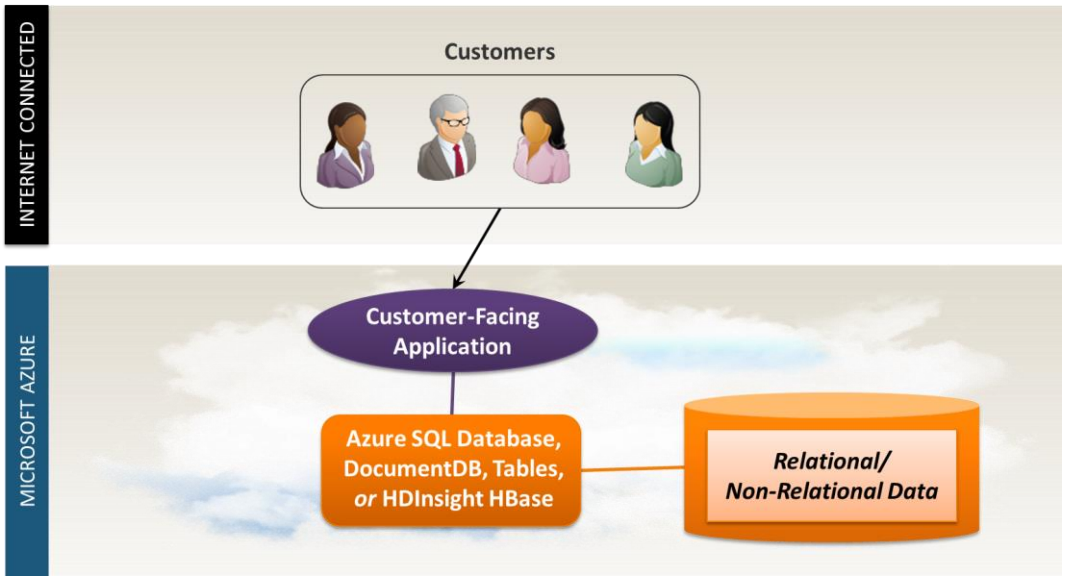

Figure 4: A new customer-facing application can use SQL Database, an Azure NoSQL technology, or both.

In some cases, it can make sense for a single application to use multiple NoSQL technologies or to combine a relational service such as SQL Database with a NoSQL service. Azure supports this approach, commonly called *polyglot persistence*, by providing these diverse services for operational data under a common billing umbrella.

Understanding Your Options

If you're creating a new customer-facing application, the first challenge is to determine whether to use a relational database, a NoSQL store, or both. If you choose to use NoSQL, you must then decide which NoSQL technology is the best fit for the application. The things to think about include these:

- SQL Database provides a standard relational service. If your application needs full relational functionality, this cloud database is the best choice. SQL Database is also a good choice if you plan to use relational technologies such as SQL Server Analysis Services to analyze the data later. And if the application's developers already know SQL, going this route can require the least training. Despite the justified attention given to NoSQL today, relational technology is still the best choice in many situations.

- DocumentDB stores everything as JSON, and so it can be the right option for applications that work with data that's already in this format. Think about a modern application, for example, with browser and mobile clients. All of those clients might send and receive data as JSON, and the application's logic might be written in

[1] Many third-party data technologies are also available on Azure. The options include managed relational services based on MySQL, NoSQL services based on MongoDB, and others.

JavaScript running on Node.js. Storing data as JSON in DocumentDB can make creating and maintaining this application significantly simpler than it would be with a relational database.

- Tables provides a fast, low-cost way to store and access unstructured data, which makes it a good choice for storing large amounts of simple data. For example, an application might create a database in SQL Database for each customer, then use Tables to store large amounts of logging and performance information generated by the application itself.

- HDInsight HBase is part of Microsoft's HDInsight offering, which provides Hadoop technologies and more as a service. A project that uses other Hadoop technologies to analyze data, such as Hive, might choose to use HBase for operational data because all of them run on a common foundation. This simplifies management, billing, and moving data among the components. HDInsight HBase is also a fast and scalable system, making it a good choice for working with non-relational data that doesn't require complex transactions.

These NoSQL technologies have another advantage, too: None of them requires using a fixed schema. This can make changes easier, an important consideration for a customer-facing application that can't have a maintenance window. Still, a fixed schema prevents some kinds of developer errors, and so choosing NoSQL can require more diligence from the development team creating this application.

NoSQL technologies have entered the mainstream—no data platform is complete without them. While they're not always the right choice, you should probably at least consider them whenever you're creating a new customer-facing application.

Azure Search

Rather than working your way through an application's menus, wouldn't you rather interact with the application through a search box? Many people prefer this, and so giving users a search option is a good thing. Yet building search into an application can be hard, requiring the application's developers to install and maintain their own search service. The result is that search boxes aren't as common as they should be in application interfaces today.

The goal of Azure Search is to change this. By providing search as a managed service on Microsoft Azure, this technology makes it easier for developers to offer this feature in their own application's user interface. It also gives them control over their searches, including the ability to specify the order in which search results are returned.

Azure Search is a little hard to categorize. It commonly works with operational data, but it doesn't address the same problems as SQL Database or a NoSQL technology. However you think about it, though, the ability to add search to an application is quite useful, and so Azure Search is an important part of the Microsoft data platform.

Scenario: Creating a Multi-Tenant SaaS Application

For many ISVs, their market demands a software-as-a-service (SaaS) offering. This SaaS application is likely to be multi-tenant, with many customers sharing the same code. The operational data challenge is to support multi-tenancy in the cloud, providing isolated data for each customer at a reasonable cost.

Describing the Scenario

Since multi-tenancy requires keeping each customer's data separate, it would be helpful if a cloud database service provided this separation for you. SQL Database does exactly this, as Figure 5 shows.

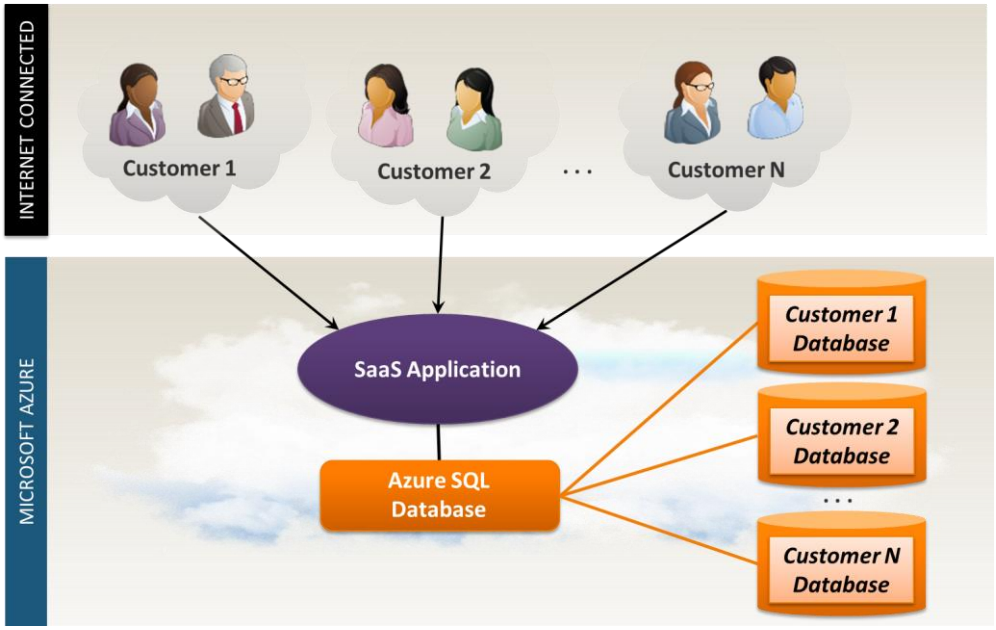

Figure 5: A multi-tenant SaaS application can use SQL Database to maintain a separate database for each customer.

As long as no single database exceeds the technology's maximum size, a SaaS application can use SQL Database to manage data in a straightforward way: just give each customer its own relational database.

Understanding Your Options

One relational database per customer is a simple and attractive solution. But a potential challenge with this approach is cost. SQL Database pricing is based on the throughput available for a database, not on how much data it stores. Three different throughput tiers are available, and a user of this service chooses which level to buy for each database. If an application's throughput demands exceed the tier's limits, the application is throttled.

To provide good performance for all of its customers, an ISV might choose the highest throughput tier—and thus the highest price—for every database. Yet usage across those databases will probably vary, with different customers active at different times. Is an ISV forced to pay for the maximum throughput for every database all the time?

Fortunately, the answer is no. SQL Database includes an option called *elastic database pools*, which lets throughput—and costs—be shared across a group of databases. Doing this lets an ISV have many high-throughput databases, one per customer, without breaking the bank. Elastic database pools can be useful in various situations, but they're especially important for creating cost-effective multi-tenant SaaS applications.

Conclusion

A modern data platform needs to address a wide range of scenarios. It must also support working with all kinds of data, including operational data, analytical data, and streaming data. Achieving these things is the goal of the Microsoft data platform.

For operational data, the platform provides a variety of solutions: on-premises and cloud, SQL and NoSQL. These technologies can be used independently or together, depending on what your applications require. They can also be combined with the platform's technologies for working with analytical and operational data. The result is a broad set of products and services for addressing a broad set of requirements.

About the Author

David Chappell is Principal of Chappell & Associates (http://www.davidchappell.com) in San Francisco, California. Through his speaking, writing, and consulting, he helps people around the world understand, use, and make better decisions about new technologies.