Microsoft SQL Data Services – Under the Hood

Writer: Jason Lee (Content Master)

Technical Reviewer: Alan Bush (Microsoft)

Published: October 2008

Applies to: SQL Data Services

Summary: SQL Data Services (SDS) is a cloud-deployed database service from Microsoft. SDS provides a web-facing database, retrieval, and manipulation features in a hosted, Web-facing solution. Cloud-deployed database solutions such as SDS can provide many benefits to the enterprise, including rapid provisioning, cost-effective scalability, high availability, and reduced management overhead. This paper provides an architectural overview of SDS, and describes how you can use SDS to augment your existing on-premises data infrastructure.

Copyright

Contents

Introduction

Companies providing Internet-based applications are facing many challenges today. Users increasingly expect access to rapidly expanding amounts of data from anywhere, at any time, and from any device. The size, scale of use, and variety of forms of data are expanding rapidly. Developers need to rapidly build and deploy applications to keep up with these growing demands. Using the traditional on-premise data management model, meeting these needs demands constant investment in and management of servers, operating systems, storage, and networking. IT and operational staff must constantly monitor the infrastructure to ensure that capacity, performance, and availability are maintained as data volumes and user loads increase.

Cloud database services, such as SDS, provide an improved way to respond to these challenges. SDS is built on three key tenets: storage for all data types from birth to archiving, rich data processing services, and operational excellence.

From a developer's perspective, SDS offers a simple programming model, using standard Internet protocols, and simple deployment options. SDS simplifies the process of creating, prototyping, and deploying applications that integrate data across the enterprise. SDS removes infrastructure obstacles, thereby giving developers more freedom to innovate and experiment with new ways of sharing data.

From the IT management perspective, SDS offers a systematic and secure cloud-deployed solution that integrates with your on-premise assets and gives the IT organization oversight and control of distributed data assets. SDS is built on the same SQL Server technologies already used and proven in on-premise deployments to provide high availability, reliability, and security.

From the business perspective, SDS offers a cost-effective approach for managing data, with a flexible consumption based pricing plan, near zero capital and operational expenditures, and the ability to quickly and easily scale up or down as your needs change.

If you are planning to build applications on large or shared data sets, provide on-demand scalable data storage, or augment your on-premise data infrastructure with low cost, rapidly provisioned cloud-based storage, SDS can provide a robust and cost-effective solution.

SQL Data Services are an essential part of Microsoft’s Cloud Services strategy and architecture, as illustrated in Figure 1.

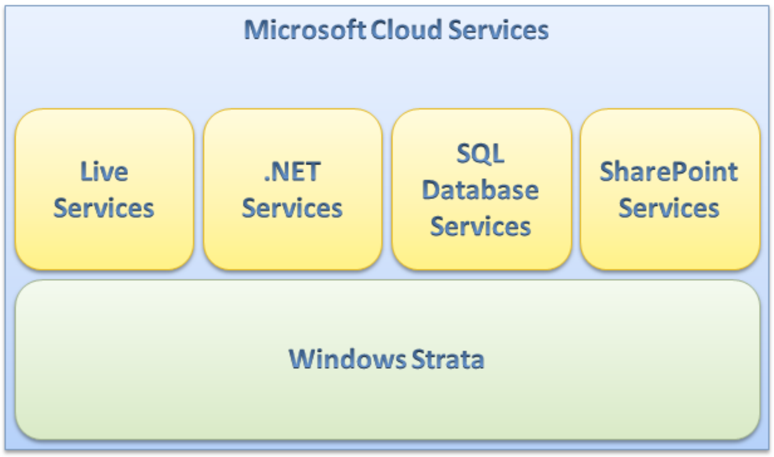

Figure 1: Microsoft Cloud Services

Key Features

SDS offers a valuable supplement to on-premise data management models in the areas of flexibility and scalability, reliability and security, and developer agility. Let's begin by looking at some of these features.

Fast provisioning

When you use the traditional on-premise data infrastructure, your ability to prototype or roll out new data-driven solutions can be slowed by the time it takes to deploy and secure servers, network components, and software. With a cloud-deployed solution such as SDS, you can provision your data storage needs in minutes.

Flexible development model

SDS exposes Web services that you can use to create, modify, query, and delete data entities. The SDS Web services currently support the two primary standards-based communications protocols, SOAP (Simple Object Access Protocol) and REST (Representational State Transfer). This ensures that you can interact with SDS from almost any programming environment, not just Microsoft-based tools, and provides cross-platform compatibility to support existing solutions and mashups. Data retrieval is accomplished through a simple, text-based query syntax using the Language Integrated Query (LINQ) style. SDS is also compatible with ADO.NET Data Services, and exposes an ADO.NET Data Services interface.

Flexible data model

SDS provides a flexible data model that does not require schemas or fixed data types. Rather than creating database tables with fixed columns, in SDS you can create data containers that can store either homogeneous or heterogeneous data entities. This flexibility gives you the ability to handle any data scenario, regardless of whether you're storing schematized relational data or totally unstructured collections of properties. This model is especially useful when you design applications in which many different users will be providing data (such as content management or collaboration), or when you create applications that many different people will use, each of whom will have somewhat different data needs (such as customer relationship management).

Unlimited scalability

SDS enables you to store any amount of data, from kilobytes to terabytes. The service scales with your data, so you can allow your data to grow without worrying about capacity limitations. A pay as you grow pricing model ensures that you only pay for the storage you use, so you can also scale down the service when you don't need it. When compared to a hosted or on-premise solution, this means that your utilization is always optimal (never over-utilized so you can't meet demand, and never under-utilized so you are paying for inefficiency).

You can harness this scalability to build the next generation of Internet-scale applications with worldwide reach, but without the infrastructure costs and management overhead.

High availability

SDS is built on robust and proven Microsoft Windows Server® and SQL Server technologies, and is flexible enough to cope with any variations in usage and load. The service stores multiple copies of your data to ensure data availability and business continuity. Published service level agreements (SLAs) will guarantee a business-ready service.

Secure data access

SDS offers secure data storage, together with secure access over Secure Sockets Layer (SSL) channels, thereby enabling you to meet business and regulatory requirements for confidentiality and privacy.

Synchronization and offline scenarios

SDS integrates with the Microsoft Sync Framework to support occasionally-connected synchronization scenarios. For example, using SDS and the Sync Framework, on-premise applications and client devices can synchronize with each other via a common data hub in the cloud.

Typical Scenarios

To see how these key features of SDS can benefit organizations, let’s consider some common business application scenarios.

Data Hub

In a data hub scenario, you typically want to enable different mobile and remote users to collaborate using the same set of data. Consider an insurance company that has a large mobile sales force consisting of more than five thousand people spread out across North America. Keeping customer and pricing data synchronized across the entire sales force is a constant problem. The first part of the problem is getting new customer contact information from the sales force into the internal finance systems. The second part is getting new pricelist information out to the sales force. The insurance company needs a solution that will:

- Keep each salesperson's laptop up to date with the latest pricing information.
- Keep the corporate system up to date with new customer information from the laptop of each salesperson, without the risk of exposing critical corporate data.

Currently, product and customer data is stored in a central SQL Server database in the data center. In addition, employees in the sales team use an application that runs on their laptops and stores data in SQL Server Express. The IT department does not want to open the firewall to the on-premise data center to provide possibly insecure access from each salesperson's laptop. The development team can provide a safe and fully synchronized solution that uses SDS, by completing the following three tasks:

1. Create a data structure in SDS to store product data and customer data.
2. Create a Microsoft Sync Provider for the data center. This Sync Provider will keep product and customer data synchronized between the data center and the SDS data hub.
3. Create a second Microsoft Sync Provider for the sales team's laptops. This Sync Provider will keep product and customer data synchronized between the field sales people and the SDS data hub.

The following diagram illustrates this solution.

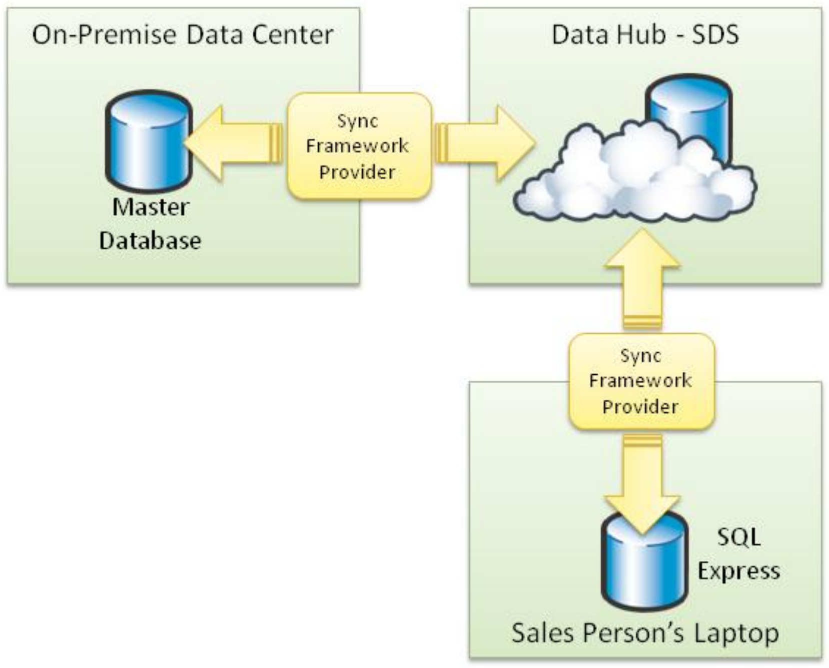

Figure 2: Conceptual overview of a data hub scenario

Product pricing data flows from the enterprise database, through SDS, to more than five thousand sales people. Customer contact data flows from more than five thousand sales people, through SDS, to the enterprise database. When a sales person's laptop is offline, changes made to local data are tracked. When the laptop's Internet connection is restored, the Sync Provider enumerates these changes and sends them to SDS. The safety of the corporate data center is ensured.

Archival and Compliance

Many organizations need to store historical data, such as financial records, business transactions, or correspondence, either for future reference or for compliance with record-keeping regulations. Consider a software-as-a-service (SaaS) vendor that provides compliance support to businesses, including financial, government, healthcare, real estate, and franchising companies. They employ a document management system for archived data that provides full text search, together with workflow functionality and check processing. They must also track and report access to resources as required for audit purposes. To lower data storage costs and help ensure rapid and secure access to records, the company would like to migrate their customers' archive data to the cloud.

Towards this aim, the company can create an account with SDS, together with .NET Services accounts within the company space for each of its customers. Once a customer has an account, they can upload any form of document, such as e-mail, scanned checks, and escrow documents. Some of the documents are stored as binary large objects (BLOBs), while other documents are stored as structured data with standardized data fields.

The uploaded content is fully indexed and enables the company to execute queries against BLOB content, entity metadata, and user-defined flexible properties. Going forward, the company could implement new services and capabilities, such as Optical Character Recognition (OCR) scanning or workflow processes built using .NET services integrated with SDS.

Architectural Overview

Technology Stack

SDS is implemented as a mid-tier application. The service exposes SOAP and REST endpoints for communication with client applications. Behind the scenes, SDS uses ADO.NET to communicate with the underlying SQL Server-based platform. The following diagram shows a conceptual illustration of the technology stack.

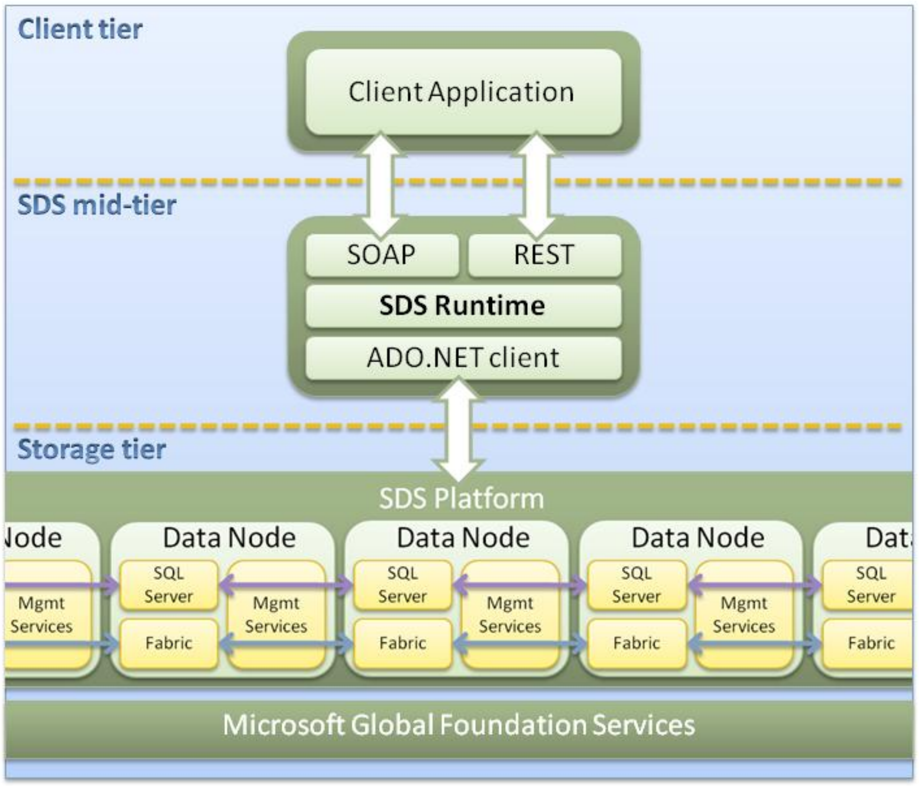

Figure 3: The SDS technology stack

To use SDS, you first provision an account. Within each account, SDS provides a simple, hierarchy-based data model that consists of three layers: authorities, containers, and entities. An authority is a collection of containers, and a container is a collection of entities. This is known as the ACE model.

The ACE Model

The following table shows the definition and purpose of authorities, containers, and entities.

Business Logic Layer	Definition	Purpose	SQL Server Analogy
Authority	Set of containers	Groups containers for accounting, security, and co-location	A SQL Server instance
Container	Set of entities	Groups entities for content and queries	An individual database
Entity	Scalar property bag	A unit of storage	An individual record

Table 1: The ACE model

An authority is the top-level organizational element in the SDS data model, and it's the first thing you create when you use SDS. Every authority is assigned a domain name system (DNS) name, which enables you to access your authority through the REST protocol and effectively makes the authority a unit of geo-location. You can think of an authority as being analogous to an individual SQL Server instance.

Each authority can contain any number of containers. You can think of a container as being analogous to an individual database. However, while a traditional database uses tables and schemas to organize data, containers impose no such restraints. You can use containers to store homogeneous data (entities of the same kind and shape) or heterogeneous data (entities of different kinds and shapes). Instead of tables and schemas, you use properties of the entities themselves to provide as much or as little structure as your solution requires.

A container can contain any number of entities. An entity includes two types of properties: flexible properties and metadata properties. Flexible properties are defined by the user and stored in a simple string object dictionary, and they enable you to define data fields in any supported data type. Currently, SDS supports **String**, **Base64Binary**, **Boolean**, **Decimal**, and **DateTime** types. These types are compatible with the types used by SQL Server.

Metadata properties are standard properties that every entity has. You can use them to structure your data and refine your queries. The following table describes the purpose of each metadata property.

Metadata Property	Purpose
ID	Provides a unique identifier for each entity.
Version	Maintains a version number for each entity. Predominantly used to synchronize cloud data from SDS with local data sources.
Kind	Allows applications to assign an arbitrary type to each entity. The interpretation of the type property is currently application specific. This property enables you to query specific kinds of entities from heterogeneous containers, or to divide the entities in a container into different kinds for special processing.

Table 2: Entity metadata properties

For example, suppose you create a container to store data on cars for sale. You might use the **Kind** metadata property to distinguish between new cars and used cars. You then could add flexible properties as required to identify body types, engine sizes, and so on.

Functional Overview

The SDS API

One of the key design goals for SDS was to support programming from any development environment. To enable this support, the service currently exposes SOAP and REST endpoints. The SOAP protocol is familiar to many developers who consume Web services, is language and platform independent, and is available in any development environment that provides access to a SOAP stack. SDS is also very well supported by the Microsoft Visual Studio® tools. Developers typically use SOAP when developing for a Microsoft-based environment, especially in enterprise applications where security and interoperability are important.

REST, on the other hand, is a lightweight, HTTP-based protocol that uses URIs and HTTP verbs to facilitate the exchange of data. As such, you can use the REST protocol from any development environment that provides access to an HTTP stack. Developers typically use REST when programming on a non-Microsoft platform, and when simple security and interoperability approaches are sufficient.

SDS has a natural REST implementation, because the principal HTTP methods map well to the key CRUD (Create, Retrieve, Update, and Delete) database operations. The following table illustrates this mapping.

HTTP Verb	SDS Operation
POST	Create
GET	Fetch, Query
PUT	Update
DELETE	Delete

Table 3: Mappings between HTTP verbs and SDS operations

When you create an authority, a DNS record is created to facilitate the use of the REST protocol. For example, if you created an authority named **london**, you would access your authority at the following URI.

```
https://london.data.beta.mssds.com/v1/
```

You can scope operations at any level of your data structure by building on this root URI.

```
https://london.data.beta.mssds.com/v1/<container-id>
```

```
https://london.data.beta.mssds.com/v1/<container-id>/<entity-id>
```

SDS will also support REST-based access using Atom Publishing Protocol (AtomPub) and JavaScript Object Notation (JSON) through the ADO.Net Data Services ("Astoria") conventions.

Manipulating Your Data

SDS exposes service methods at every level of the data model. Regardless of whether you use SOAP or REST, each operation follows the same pattern: you set the scope of the operation (service, authority, container, or entity) and call the appropriate method. The following table summarizes the service operations that are available at each scope.

Scope	Service Operations
Service	▪ CreateAuthority
Authority	▪ FetchAuthority ▪ QueryContainers ▪ CreateContainer
Container	▪ FetchContainer ▪ QueryEntities ▪ DeleteContainer ▪ CreateEntity
Entity	▪ FetchEntity

	▪ UpdateEntity ▪ DeleteEntity

Table 4: SDS scopes and service operations

If you use the SOAP protocol, you create a Scope object and pass this object as a parameter when you call the appropriate method. If you use the REST protocol, the URI represents the scope of the operation. You create an HTTP request to the URI, call the appropriate HTTP method, and write your data to the request stream as an XML payload.

Querying Your Data

SDS uses a simple text-based query language, based on the LINQ pattern for C#.

You can scope queries to either an individual authority or to an individual container:

- You can query an authority for the containers within it that match a specified condition.
- You can query a container for the entities within it that match a specified condition.

An SDS query takes the following format:

```
from e in entities [where condition] select e
```

Note that the **from e in entities** format applies regardless of whether you are retrieving containers or entities. In SDS, authorities, containers, and entities are all types of flexible entities.

To submit your query, you must specify the scope and then supply the query expression. For example, to query for containers by using the REST protocol, you simply append your query to the authority URI as a query string:

```
https://london.data.beta.mssds.com/v1/?='from e in entities…'
```

Similarly, to query for entities, you append your query to the container URI:

```
https://london.data.beta.mssds.com/v1/<container-id>/?='from e in entities…'
```

You can use comparison operators and Boolean operators to add conditions to your queries. The current beta version of SDS supports the following operators:

Comparison operators	> (greater than) >= (greater than or equal to) < (less than) <= (less than or equal to) == (equal to) != (not equal to)
Boolean operators	&& (logical AND) || (logical OR) ! (logical NOT)

Table 5: Supported comparison and Boolean operators

For example, to retrieve all the entities within a container that have a **City** property equal to **Seattle** and the **State** property equal to **WA**, you would structure your query as follows:

```
from e in entities where e["City"] == "Seattle" && e["State"] == "WA" select
e
```

To query metadata properties, you use the dot notation instead. For example:

```
from e in entities where e.Id == "exampleID" select e
```

Simple join operations are also supported. The current query capabilities of SDS are a first step, and will be extended over time.

Securing Your Data

SDS provides security at the account and authority levels. Accounts are secured by a Windows Live™ ID. Each authority is secured by a single account name and password combination that grants read/write access. Soon SDS will also have integration with Microsoft Windows® .NET Access Control Services for richer authentication and more granular levels of authorization.

BLOB Support

We asked architects and developers which features they would most like to see included in a flexible data infrastructure, and BLOB support was at the top of the list. BLOB support enables you to store videos, images, and other files as binary-encoded data, making SDS a feasible data platform for photo or video clip libraries, music players, document stores, and any number of other innovative applications. The current beta version of SDS supports BLOBs of up to 100MB, and upcoming releases will support larger BLOBs.

In SDS, BLOBs are defined in a similar way to other entities. Each BLOB entity has **Id**, **Version**, and **Kind** metadata properties, although currently the **Kind** property has a fixed value of "Entity". However, BLOB entities have an additional metadata property named **Content**. The **Content** property has three attributes that you use to record information on how the BLOB should be handled.

Content attribute	Purpose
content-type	Specifies the MIME type of the BLOB content.
content-length	Specifies the size of the BLOB.
content-disposition	Optionally specifies how the client should handle the BLOB. For example, Web browsers use the content-disposition header to suggest a filename when a user downloads a file.

Table 6: Attributes of the Content metadata property

Currently, you must use the REST interface to work with BLOB data in SDS.

Support for ADO.NET Data Services

The ADO.NET Data Services framework is a set of patterns and libraries that you can use to create and consume Web-based data services. The framework was conceived to support two key trends.

First, Web applications increasingly rely on client-side technologies such as Microsoft Silverlight™ and Flash to drive presentation and interactivity. The traditional post-back model used to bind server-side data to client-side presentation is increasingly inadequate for these rich interactive applications. ADO.NET Data Services enables the Web browser to retrieve and consume data directly from an Internet-facing data source.

Second, Web-based mashup applications that aggregate data from many disparate sources are becoming increasingly commonplace. ADO.NET Data Services can help you to both provide and consume data for this type of application.

SDS will provide an ADO.NET Data Services-compatible interface. This interface is now available in preview form. You can direct your client software to use this interface in the same way you would use any Web service compliant with ADO.NET Data Services.

Conclusion

In this paper, we have introduced SDS and described its key capabilities and benefits. SDS is a cloud-deployed database service that offers developer agility, application flexibility, and virtually unlimited scalability, with a flexible, cost-effective delivery model. The robust underlying architecture provides reliability, high availability, and security, while support for the most prevalent Internet communication protocols ensures ease of deployment and use. We have examined some scenarios where SDS can offer real business value to customers, such as data hub solutions and archival and compliance systems. We've also taken a look "under the hood" at the architectural and programming models that provide the core functionality of SDS.

For more information, please visit:

- SQL Data Services Portal
  http://www.microsoft.com/sql/dataservices
- SQL Data Services Dev Center
  http://msdn.microsoft.com/en-us/sqlserver
- SQL Data Services Documentation
  http://msdn.microsoft.com/en-us/library/cc512417.aspx
- SQL Data Services Team Blog
  http://blogs.msdn.com/SDS