8. Particionamiento de Datos

Hay múltiples factores que pueden influir directamente en cómo se almacenan los datos para una carga de trabajo determinada en un entorno SaaS. El cumplimiento normativo, el problema del vecino ruidoso, el aislamiento, el rendimiento y el costo —cualquiera de estos puede tener una influencia significativa en cómo se decide representar los datos en un entorno multi-tenant.

Fundamentos del particionamiento de datos

Cuando pienso en almacenar datos en un entorno multi-tenant, siempre considero cómo se particionarán los datos de cada tenant en función del tipo de datos que se almacenan, la tecnología que se utilizará para almacenarlos y gestionarlos, cómo los consumirá el tenant, y otros factores similares.

El particionamiento de datos no presupone que los datos de cada tenant deban ubicarse necesariamente en un elemento de almacenamiento completamente independiente. De hecho, es aquí donde los conceptos de silo y pool vuelven a desempeñar un papel clave.

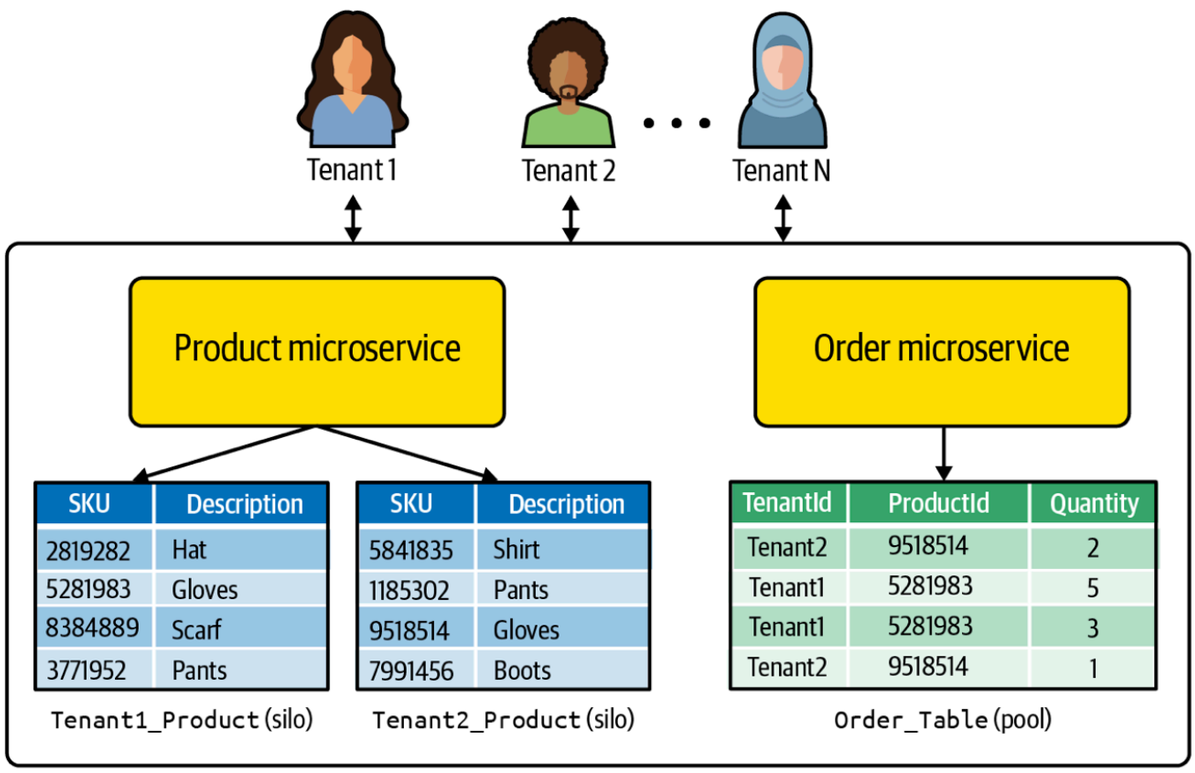

Figure 8-1. Siloed and pooled data partitioning models

La clave es que, independientemente de la tecnología utilizada para almacenar los datos, estos términos —silo y pool— siguen siendo válidos para caracterizar la forma en que se representan los datos.

Cargas de trabajo, SLA y experiencia

Toda estrategia de almacenamiento multi-tenant debe estar muy enfocada en la escalabilidad y el rendimiento.

La naturaleza del consumo de datos multi-tenant puede variar significativamente entre los servicios de la solución, lo que requiere enfoques creativos para soportar las cargas de trabajo cambiantes y los patrones de consumo de los tenants.

Parte de este ejercicio también consiste en estimar la huella de recursos de los datos. Tener una idea del volumen de datos que los tenants almacenarán en los distintos servicios del sistema permite proyectar cómo podría escalar el sistema para cumplir con los SLA de los tenants.

El problema del vecino ruidoso es otro aspecto en el que conviene enfocarse al construir la estrategia de almacenamiento multi-tenant. Ya se ha abordado como una preocupación de carácter global, pero este problema tiene sus propios matices específicos en el contexto del almacenamiento multi-tenant.

Radio de impacto

Si bien el objetivo de toda arquitectura es hacer todo lo posible para prevenir interrupciones del servicio, hay escenarios en los que los equipos adoptan medidas adicionales para garantizar que un tenant no pueda afectar la experiencia de otro.

Al elegir un modelo de almacenamiento, algunos equipos incorporan el radio de impacto en su estrategia general, inclinándose hacia el siloing de datos para reducir el alcance de posibles interrupciones. Esto puede traducirse en aislar en un silo una familia de datos especialmente crítica.

Si bien el siloing de datos para limitar el radio de impacto tiene sus ventajas, es algo que debe evaluarse con cuidado. Puede ser adecuado en algunos entornos, pero también puede convertirse en un recurso que afecte la agilidad general, la eficiencia de costos y el perfil operacional del entorno.

La influencia del aislamiento

El aislamiento toca todas las dimensiones de la arquitectura multi-tenant y, sin duda, juega un papel en la elección del modelo de particionamiento de datos. **Aunque se haya optado por un modelo silo para habilitar el aislamiento, el siloing de los datos por sí solo no implica que los datos estén aislados.**

Es preferible entender el aislamiento como un factor que influye en la elección del modelo de particionamiento de datos, reconociendo que la implementación efectiva del aislamiento se logra de forma separada. Esta influencia del aislamiento sobre el particionamiento de datos cobra especial importancia cuando se analizan diferentes tecnologías de almacenamiento.

Gestión y operaciones

Como arquitectos y desarrolladores SaaS, es natural concentrarse en el rendimiento y el aislamiento al elegir una estrategia de particionamiento de datos. Sin embargo, es igualmente importante considerar cómo dicha estrategia influirá en la gestión y la huella de recursos operacional del entorno SaaS.

El modelo de particionamiento de datos pooled, por razones obvias, suele ofrecer el mayor nivel de eficiencia operacional. Cuando los datos están representados en un modelo pooled, la historia de gestión y operaciones es mucho más clara.

Con el particionamiento de datos siloed, se trabaja con un modelo más distribuido que añade cierta complejidad a la gestión y las operaciones. Aquí, las herramientas de DevOps deben afrontar el desafío de aplicar cualquier cambio de datos o migración a cada tenant de forma individual. Sincronizar estos cambios en una gran colección de elementos de datos siloed agrega un nivel considerable de complejidad y coordinación temporal a este proceso.

El respaldo y la restauración también deben formar parte del análisis. ¿Cómo influirá el modelo de particionamiento en la capacidad de respaldar y restaurar los datos de un tenant? Con datos siloed, esto resulta más sencillo. Sin embargo, con datos pooled es necesario pensar cómo se tomaría una instantánea de los datos de un tenant desde un servicio de almacenamiento compartido, y considerar cómo se restaurarían esos datos sin afectar a los demás tenants.

La herramienta adecuada para cada tarea

Algunos equipos ven el particionamiento de datos como una decisión de todo o nada, en la que se elige una tecnología que satisfaga los requisitos de la solución.

Si se concibe el sistema como una colección de servicios que encapsulan su almacenamiento subyacente, entonces las opciones de tecnología de almacenamiento deben evaluarse servicio por servicio.

El objetivo es considerar cada servicio de forma independiente, permitiéndole ofrecer la mejor experiencia posible en función de la carga de trabajo que soporta, sus necesidades de aislamiento, sus consideraciones de cumplimiento normativo, su perfil de gestión y todos los demás factores que puedan determinar su huella de recursos.

Esto incluye elegir estrategias de particionamiento siloed y pooled servicio por servicio. Es probable que se identifiquen patrones de almacenamiento que se apliquen universalmente a muchos servicios; sin embargo, conviene formular estas preguntas para cada servicio que se construya.

El modelo pooled como punto de partida

En general, la recomendación es adoptar el modelo pooled como punto de partida para todo el almacenamiento del sistema. Las ventajas en costo, operaciones y agilidad del almacenamiento pooled suelen ser tan convincentes que los equipos tienden a mantenerlo el mayor tiempo posible.

Con este enfoque, se exige que cualquier dato siloed justifique su separación mediante un conjunto claro de requisitos que avalen asumir los compromisos que implica gestionar, operar y desplegar datos en un modelo siloed.

Cada vez que se decide aislar un recurso en un entorno SaaS, conviene cuestionar los supuestos y asegurarse de haber definido el silo en el límite adecuado.

Soporte para múltiples entornos

Cada vez que se crea uno de estos elementos específicos de tenant, es necesario asignarles un nombre que lo asocie con el tenant correspondiente. **Resolver esto generalmente implica definir una convención de nomenclatura que garantice que el recurso sea identificado de forma única.**

El asunto se vuelve más interesante cuando se considera el soporte para múltiples entornos (QA, staging, producción). En ese caso, es necesario determinar si la convención de nomenclatura debe incluir una referencia al entorno que aloja el recurso de almacenamiento siloed.

Para algunos servicios y estrategias, puede existir un elemento de siloing que no genere conflictos entre nombres. Por ejemplo, en algunos servicios de almacenamiento en la nube puede haber cuentas separadas por entorno. En ese modo, los nombres de los recursos estarían delimitados al nivel de la cuenta, garantizando que solo sean accedidos desde dentro de esa cuenta.

El desafío del dimensionamiento adecuado

Un tema recurrente a lo largo de este libro es la noción de alinear el consumo de infraestructura con la actividad de los tenants. Todo esto forma parte de la búsqueda de eficiencia que es fundamental para todo negocio SaaS.

El dimensionamiento del cómputo de almacenamiento es más complicado en un entorno multi-tenant. Con los servicios, es posible escalar el cómputo horizontalmente en función de la carga, lo que facilita expandir y contraer la huella de recursos de

cómputo según las cargas de trabajo de los tenants. **Sin embargo, con el almacenamiento generalmente no se dispone de esta opción. En cambio, un servicio de almacenamiento suele requerir seleccionar un perfil de cómputo específico al configurar inicialmente el entorno.** Esto significa que el cómputo de almacenamiento permanece fijo para todas las cargas de trabajo de los tenants.

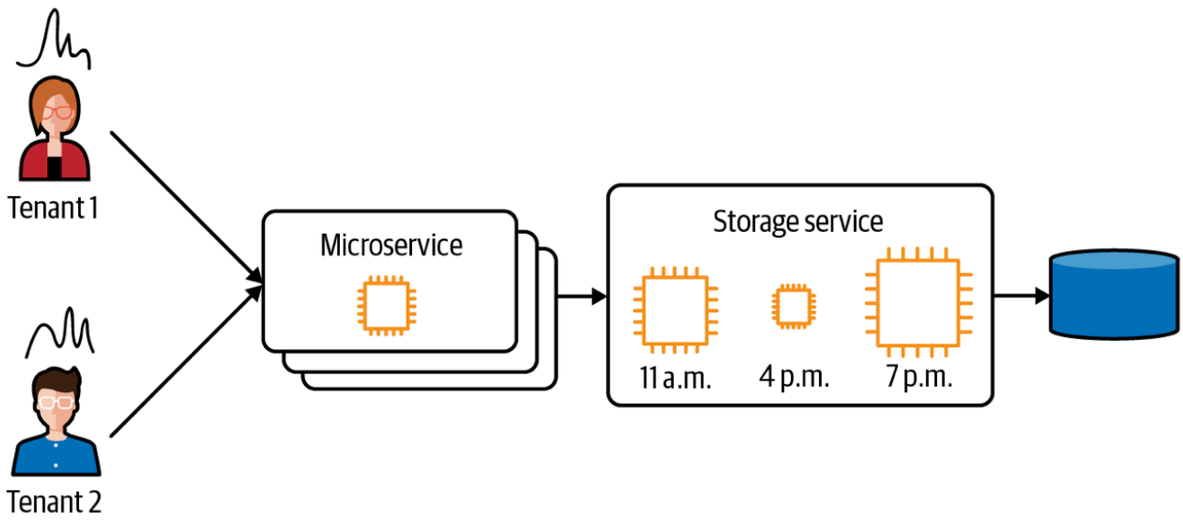

Figure 8-2. Sizing the compute of your storage

Abordar este problema se ve algo diferente en los modelos de particionamiento siloed y pooled. En el modelo pooled, es muy difícil dimensionar adecuadamente el cómputo del servicio de almacenamiento. En general, para acomodar las cargas de trabajo cambiantes y los picos de múltiples tenants que consumen el almacenamiento pooled, será necesario sobredimensionar el cómputo y aceptar que esto tendrá un impacto negativo en las estrategias de eficiencia de costos del entorno.

En definitiva, no existe una solución mágica para este desafío. Si el servicio de almacenamiento exige vincularse a un perfil de cómputo fijo, habrá que aceptar los retos que ello conlleva. En el mejor de los casos, los costos asociados al sobredimensionamiento de los recursos no tendrán un impacto significativo en los márgenes generales del negocio SaaS.

Rendimiento y regulación

Una forma de abordar los problemas de dimensionamiento adecuado es mediante el uso de estrategias de rendimiento y regulación del almacenamiento. Estos mecanismos permiten introducir políticas que gestionen mejor el consumo de los recursos de almacenamiento, incluyendo el soporte de experiencias diferenciadas según el tier y el perfil de los tenants.

La idea es usar las opciones de configuración de rendimiento integradas en la tecnología de almacenamiento para encontrar la combinación de ajustes que mejor soporte los desafíos generales de rendimiento y dimensionamiento de las cargas de trabajo de almacenamiento.

La clave es que, como parte del modelo de particionamiento de datos, debe considerarse cómo se aplicarán estas políticas. Esto incluye pensar en cómo podrían diferir para los modelos de almacenamiento pooled y siloed.

Almacenamiento serverless

Al analizar este problema de dimensionamiento, se observa que está completamente concentrado en aquellas tecnologías de almacenamiento que dependen de predimensionar su huella de recursos de cómputo.

Con el almacenamiento serverless, estos servicios ocultan los detalles del perfil de cómputo de almacenamiento, dimensionando el cómputo subyacente en función del nivel actual de actividad de los tenants. El cómputo se dimensiona automáticamente según la carga del sistema, lo que encaja perfectamente con las necesidades de un entorno multi-tenant.

En AWS, por ejemplo, se encuentra Amazon Aurora Serverless, un servicio de base de datos relacional serverless que emplea este modelo. Amazon DynamoDB también es serverless, proporcionando un servicio de almacenamiento NoSQL gestionado que no requiere dimensionamiento de cómputo.

En cualquier lugar donde se utilice el modelo de particionamiento de datos pooled, por ejemplo, un modelo serverless puede representar una opción muy atractiva.

Este modelo serverless también simplifica la operación y gestión de los servicios de almacenamiento, eliminando la necesidad de ajustar constantemente los requisitos de dimensionamiento de cómputo.

Particionamiento en bases de datos relacionales

Las bases de datos relacionales generalmente soportan una variedad de elementos distintos que pueden usarse para particionar los datos. Una constante es la dependencia del esquema. Esto significa que, independientemente del modelo de particionamiento de datos elegido, los datos estarán representados en un entorno vinculado a un esquema. Esto puede añadir complejidad a la historia de agilidad general del modelo de almacenamiento.

Particionamiento relacional en modelo pooled

Los mecanismos para almacenar datos en un modelo pooled con una base de datos relacional son bastante sencillos. Se toma el diseño de base de datos existente y se introduce un identificador de tenant para asociar los elementos de una tabla con un tenant

específico.

	TenantId	CustomerId	FirstName	LastName
Tenant1	5d4402a6-e759-432c-9e27-c9e1c8d1f970	84829349	Jane	Smith
Tenant2	24150732-cba3-49ad-a39e-a955e01d3ba7	64829451	Mike	Jones
Tenant1	5d4402a6-e759-432c-9e27-c9e1c8d1f970	28192826	Sue	Henderson
Tenant3	702257a5-fbdf-4890-932f-f1a4faabae3b	95185144	Joe	Michaels
Tenant4	de1115e1-cf6b-4bf2-b22b-616b247ead0e	84965872	Lisa	Lewis
Tenant2	24150732-cba3-49ad-a39e-a955e01d3ba7	20594091	Rob	Hanson

Figure 8-3. Pooled relational database

La idea básica de cualquier modelo de almacenamiento pooled es que los datos de todos los tenants se mezclan dentro de un elemento compartido. Luego, el dato que anteriormente habría sido la clave foránea de esta tabla se convierte en un índice secundario o en un filtro.

Introducir una columna TenantId también nos da la posibilidad de aplicar políticas que formarán parte de nuestra estrategia de aislamiento de tenants. **La clave es que, en algunos casos, el modelo de particionamiento de los datos pooled puede influir en las estrategias de aislamiento que pueden aplicarse a los datos.**

Particionamiento relacional en modelo siloed

Una tecnología de base de datos relacional puede ofrecer múltiples elementos para definir los límites del silo de los datos.

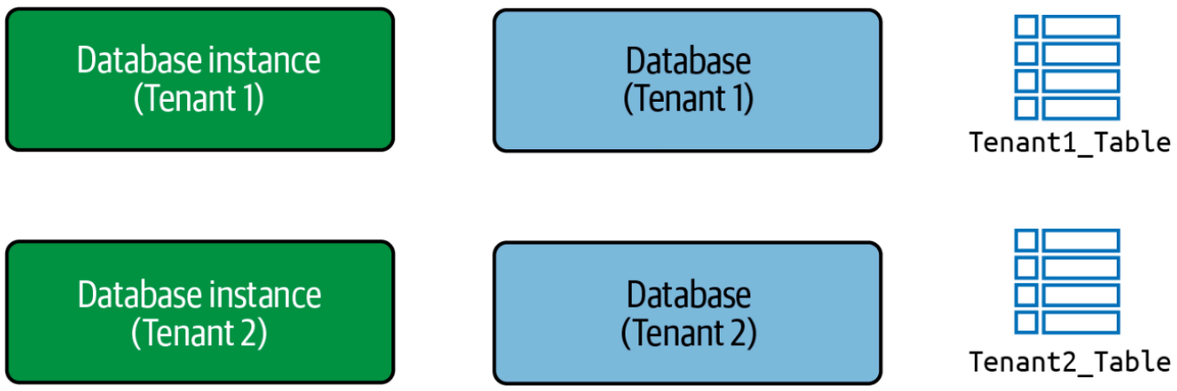

Figure 8-4. Picking a relational siloed storage construct

1. En el lado izquierdo del diagrama se muestra un modelo de instancia de base de datos por tenant. Con el servicio Amazon Relational Database Service (RDS), por ejemplo, esto equivaldría esencialmente a crear infraestructura completamente separada para cada tenant.
2. En el centro se ilustra el concepto de una base de datos. Aquí, hay una base de datos dedicada para cada tenant ejecutándose dentro de una instancia de base de datos. Aunque cada base de datos es completamente independiente, todas comparten el cómputo subyacente de la instancia de base de datos padre.
3. Por último, en el lado derecho se muestra un modelo donde se usan tablas separadas para aislar los datos de cada tenant en un silo. Todas estas tablas se crearían dentro de una base de datos compartida.

En tiempo de ejecución, el código necesitaría mapear una solicitud de tenant determinada a su instancia de base de datos, base de datos o tabla correspondiente.

Para cada uno de estos elementos de silo, es necesario considerar si el elemento puede escalar para satisfacer las necesidades. Dentro de RDS, por ejemplo, hay límites que condicionarán las opciones disponibles. Solo se puede tener un número determinado de bases de datos dentro de una instancia de base de datos.

La recomendación general es considerar cuidadosamente el rango de límites que acompañan a la tecnología relacional elegida.

Particionamiento de datos NoSQL

La naturaleza sin esquema del almacenamiento NoSQL ofrece ventajas significativas a los desarrolladores SaaS multi-tenant. Imagina gestionar un entorno pooled de gran escala y desplegar un cambio que incluye nuevas estructuras de datos.

Con una configuración relacional, sería necesario orquestar una serie de pasos complejos de migración para actualizar el esquema como parte del lanzamiento de una nueva funcionalidad.

Ese mismo cambio generalmente es mucho más fácil de aplicar en un entorno NoSQL. La ausencia de un esquema suele eliminar la necesidad de navegar por una serie de scripts de migración complejos, lo que permite a los equipos introducir funcionalidades a un ritmo más rápido y con menos despliegues.

En general, la recomendación a los equipos es comenzar con NoSQL y dejar que los factores del negocio real orienten hacia lo relacional cuando tenga sentido.

Particionamiento NoSQL en modelo pooled

El pooling de datos en NoSQL es muy similar al de una base de datos relacional. Con DynamoDB, todos los datos se almacenan en tablas. Los datos en estas tablas incluirán información de todos los tenants del sistema.

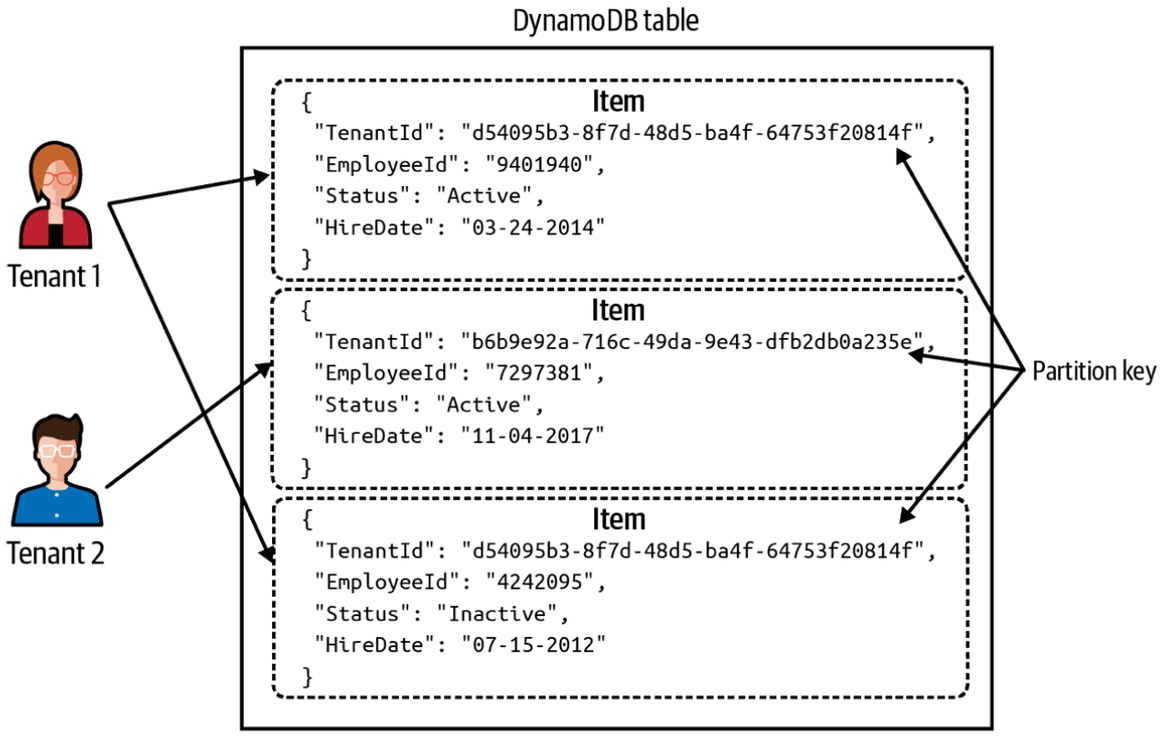

Figure 8-5. Pooled NoSQL data partitioning

Se ha indicado que cada uno de los identificadores de tenant en la tabla corresponde a lo que DynamoDB denomina partition keys. Esto esencialmente identifica el atributo como la clave primaria de la tabla, mejorando el rendimiento al acceder a los datos de tenants individuales.

Particionamiento NoSQL en modelo siloed

En muchos modelos siloed, se tiende a buscar algún mapeo lógico a una instancia de base de datos que contenga los datos del silo. Sin embargo, con DynamoDB (y otros servicios de almacenamiento gestionados), estos elementos no existen como tal. En cambio, **la única opción de silo disponible es utilizar un modelo de tabla por tenant para aislar los datos.**

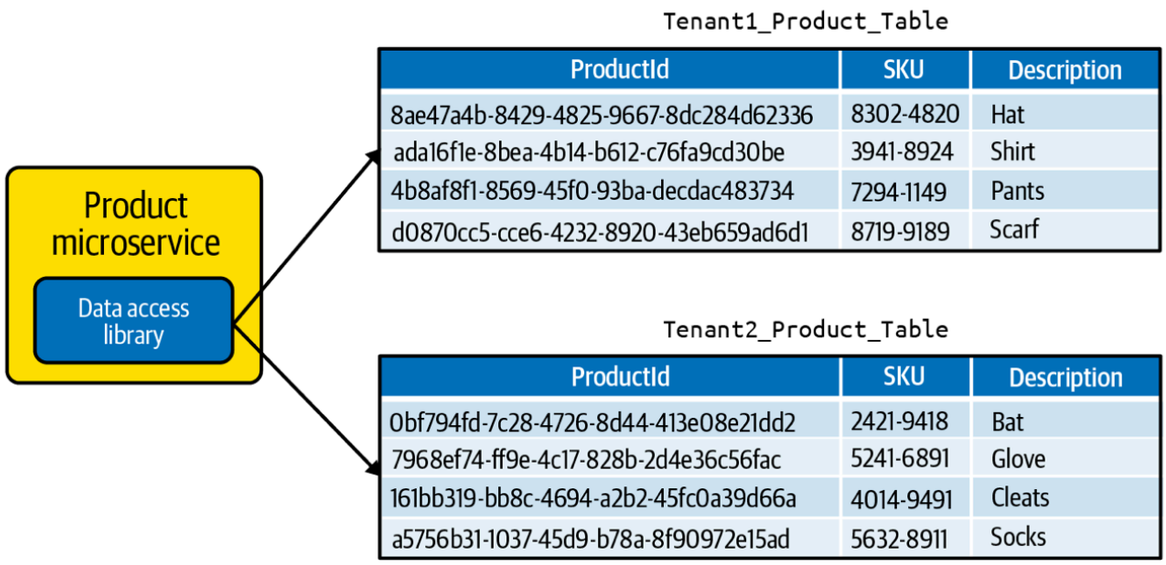

Figure 8-6. NoSQL siloed data partitioning

No hay nada exótico en el enfoque siloed con DynamoDB. Básicamente, se crea una tabla separada para cada tenant.

En este ejemplo, se añade un identificador de tenant como prefijo al nombre de la tabla para asociarla a un tenant específico. La estrategia de nomenclatura que se adopte debe elegirse con cuidado. Los nombres de tablas en DynamoDB se gestionan globalmente, lo que significa que será necesario identificar un esquema de nomenclatura que garantice que cada tabla tenga un nombre único.

Siempre que se considere un modelo de almacenamiento siloed, también debe evaluarse si la estrategia de siloing soportará los requisitos de escalado de la solución. Al igual que con las bases de datos relacionales, también hay que considerar los límites de DynamoDB (o cualquier otra solución NoSQL que se utilice).

Opciones de ajuste de NoSQL

Sin embargo, hay algunos factores adicionales que pueden entrar en juego al almacenar datos multi-tenant con DynamoDB. Por ejemplo, DynamoDB ofrece opciones para configurar los modos de capacidad de una tabla.

Con el modelo on-demand, es posible que DynamoDB escale en función de la carga real de los tenants. Esto, como puede imaginarse, encaja muy bien con la imprevisibilidad del consumo de tenants que acompaña a las cargas de trabajo multi-tenant.

El modelo provisioned es mucho más adecuado para entornos donde se tiene un mejor conocimiento de los niveles de actividad que será necesario soportar. Aquí se configuran la capacidad mínima, la capacidad máxima y la utilización objetivo en función del perfil de la carga de trabajo.

Evidentemente, al analizar otras soluciones NoSQL, pueden estar disponibles otras opciones de particionamiento y configuración. En esencia, sin embargo, todo se reduce a determinar qué elementos podrían utilizarse para representar la experiencia siloed.

Particionamiento de base de datos de objetos

Con el almacenamiento de objetos, no se analizan los datos a través del prisma de bases de datos y tablas. En cambio, los activos gestionados se visualizan como una serie de objetos que equivalen esencialmente a archivos almacenados y recuperados en diferentes contextos.

A efectos de esta discusión, me centraré en el servicio Amazon Simple Storage Service (S3), explorando los diferentes mecanismos que S3 ofrece para particionar datos multi-tenant.

Particionamiento de objetos en modelo pooled

Con el almacenamiento de objetos, el enfoque principal es utilizar una estructura jerárquica de carpetas clásica para organizar y acceder a los objetos.

Con el modelo pooled, hay que empezar por decidir dónde se ubicarán los objetos de los tenants dentro de S3. Dado que todos los objetos de los tenants estarán mezclados, no es necesario preocuparse por crear buckets o prefix keys separados para cada tenant.

Al mismo tiempo, es probable que los servicios de la aplicación necesiten operar sobre los objetos de los tenants en grupos. Esto sugiere que conviene usar las prefix keys para determinar dónde se ubican los objetos de cada tenant.

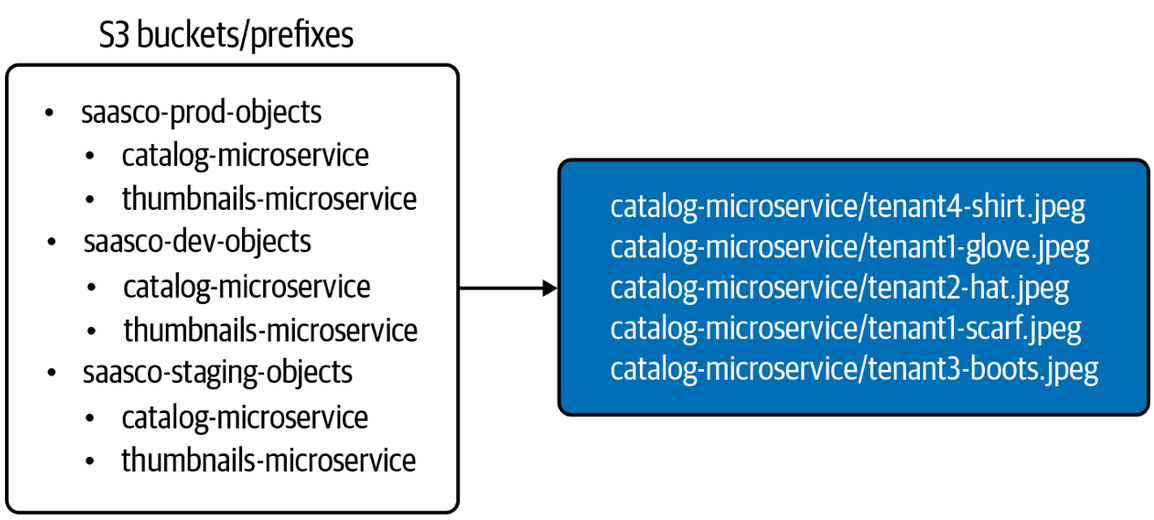

Figure 8-7. S3 pooled data partitioning

Dado que se trata de un modelo pooled, los objetos se nombran con un identificador de tenant como prefijo, que asocia cada objeto con su tenant correspondiente.

La adición de identificadores de tenant como prefijo puede resultar difícil de gestionar y añade fricción a cualquier cliente que realice llamadas. Al pasar al modelo silo, se verá cómo las ventajas de este modelo ofrecen a los desarrolladores SaaS una ligera variación del modelo pooled que elimina parte de la sobrecarga de nomenclatura y mapeo.

Particionamiento de objetos en modelo siloed

Hay dos enfoques para el siloing de objetos con S3. **La opción más sencilla es el modelo de bucket por tenant.** Este modelo requeriría crear un nuevo bucket para cada nuevo tenant y almacenar todos sus objetos en ese bucket.

Si el número total de tenants es inferior al límite máximo de buckets del servicio S3 (actualmente 1.000 buckets), esta opción puede considerarse válida.

Si estos límites representan un problema o se desea preocuparse menos por las colisiones de nombres, se puede añadir una ligera variación a la estrategia pooled (discutida anteriormente) y utilizar una prefix key como forma de implementar el modelo siloed.

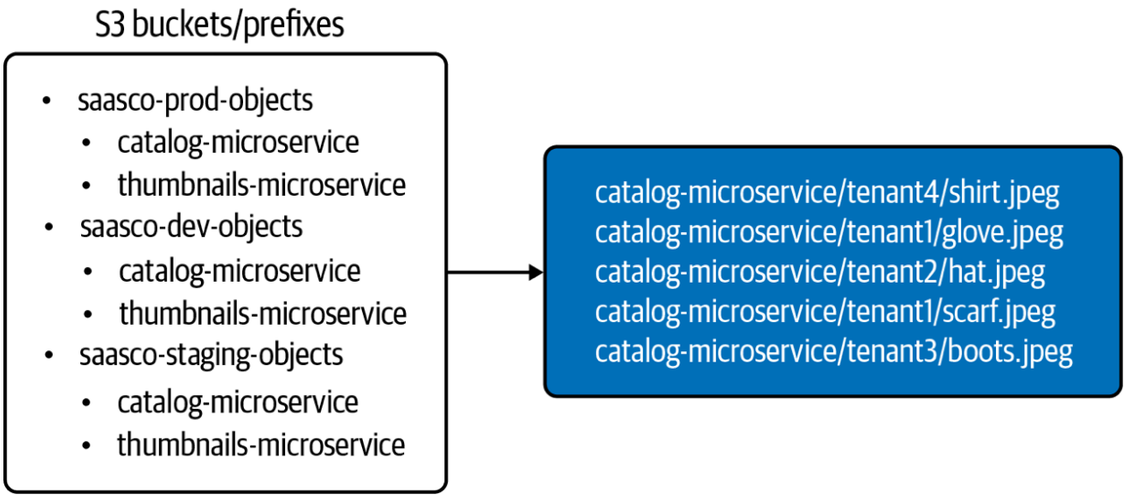

Figure 8-8. S3 siloed data partitioning

Se notará que hay una diferencia sutil entre este enfoque y el modelo pooled. Básicamente, se utiliza la prefix key como límite del silo, colocando todos los datos del tenant bajo una única clave.

Aquí, con simplemente refinar la clave, se obtiene un modelo más fácil de usar, consumir y gestionar sin asumir ningún inconveniente significativo.

Acceso gestionado por base de datos

Si bien S3 proporciona APIs para acceder a los objetos, algunos casos de uso requieren un enfoque más dinámico para localizar los objetos de los tenants. Hay instancias en las que puede ser necesario soportar formas más elaboradas, orientadas a metadatos, para localizar un objeto S3 de un tenant. En estos casos, los metadatos que se desea buscar probablemente residan fuera del almacenamiento de objetos.

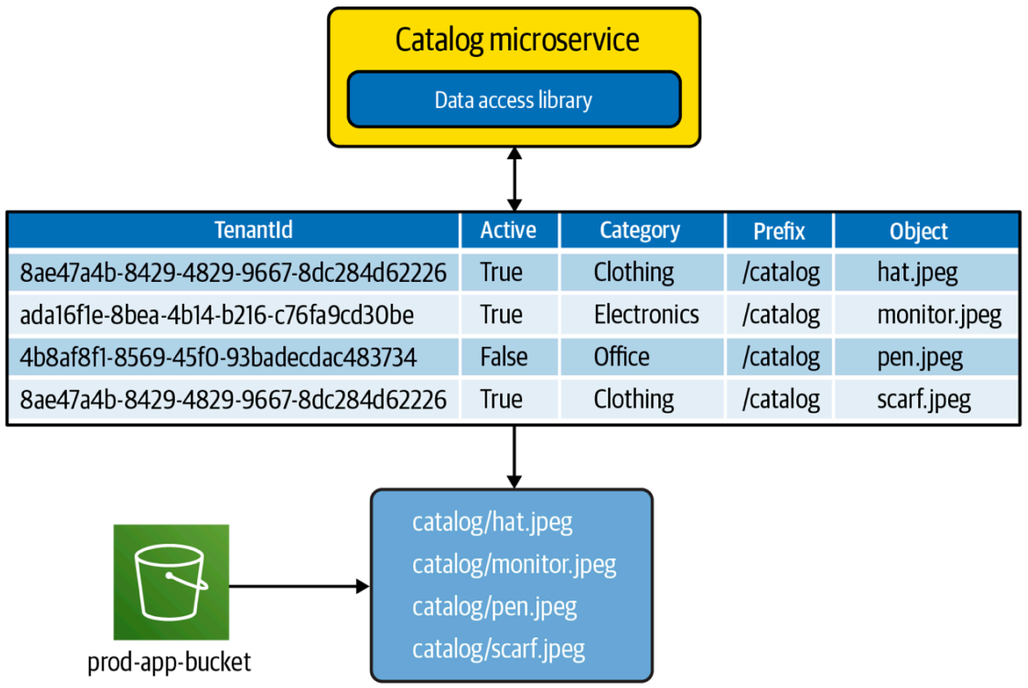

Figure 8-9. Using a database to manage access to objects

Para soportar esta experiencia, se introduce una tabla indexada por identificadores de tenant que contiene otros metadatos sobre los elementos del catálogo.

Con todos los elementos en su lugar, el servicio de catálogo puede consultar la tabla para encontrar los objetos de un tenant que coincidan con un conjunto de criterios especificados. De hecho, se puede observar que la primera y la última fila de esta tabla están asociadas al mismo tenant.

Aquí, el particionamiento se realiza completamente en la tabla, eliminando cualquier mapeo de tenant de la base de datos de objetos. Esta tabla permite introducir metadatos para los objetos que pueden aplicarse como parte del filtrado de acceso a los objetos necesarios para un caso de uso determinado.

Particionamiento de datos en OpenSearch

Vale la pena analizar cómo se aplica el particionamiento de datos en el dominio de búsqueda y analítica.

En el nivel más externo, se encuentran los dominios que representan las unidades más gruesas de particionamiento de datos. Estos son esencialmente los clusters que forman parte de la experiencia de búsqueda y analítica.

Aquí es donde se configura el tamaño y la huella de recursos de los nodos que darán forma al perfil de escalado de la experiencia. A la derecha, hay una serie de índices asociados a un dominio determinado. Estos índices almacenan los documentos que serán indexados para la experiencia de búsqueda y analítica.

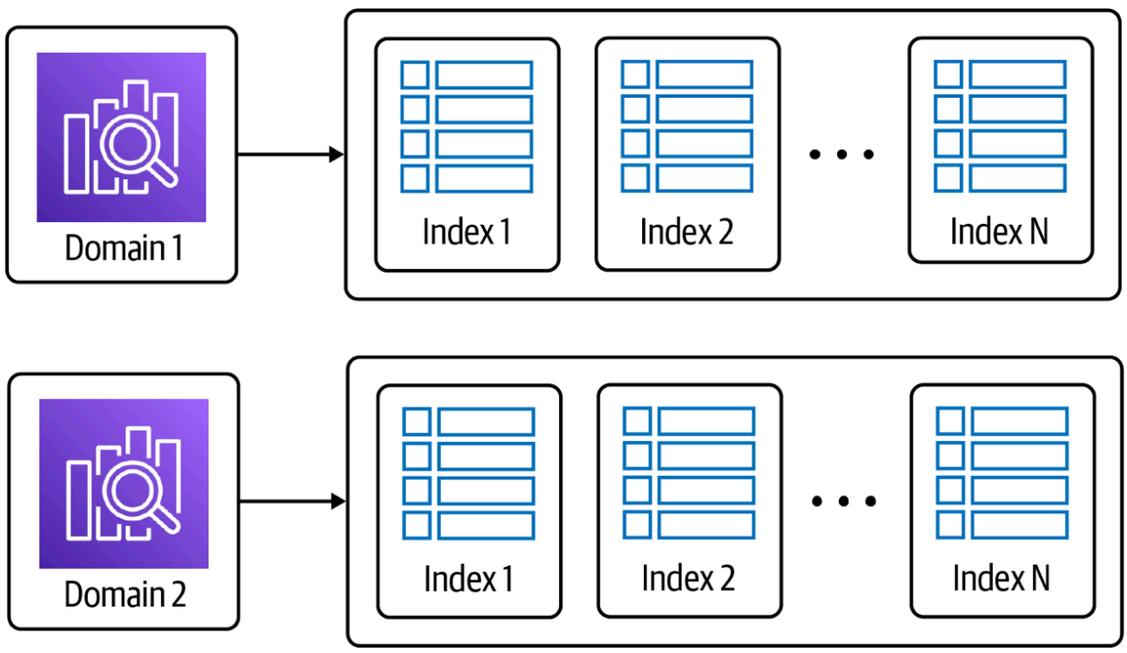

Figure 8-10. OpenSearch data partitioning constructs

Particionamiento de OpenSearch en modelo pooled

Al analizar cómo se ubican los datos en un modelo pooled en OpenSearch, es necesario adoptar un modelo donde los documentos de los tenants se mezclen dentro de un único índice. Este enfoque, en muchos aspectos, sigue el patrón observado en todos los elementos pooled.

Al examinar los distintos documentos almacenados en un índice, se observa que es necesario insertar un identificador de tenant en cada documento. Este identificador se incluirá en cualquier búsqueda que se realice para delimitar el acceso a los datos de un tenant específico.

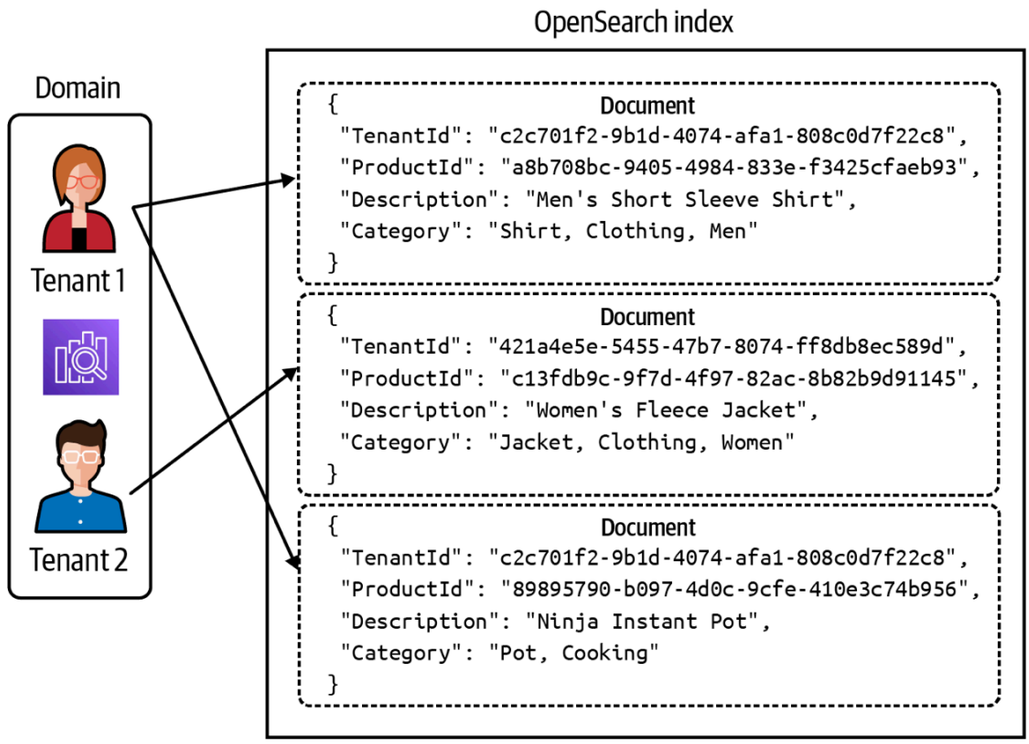

Figure 8-11. Pooled OpenSearch data partitioning

Con esta experiencia, se obtienen todas las ventajas e inconvenientes que acompañan a un modelo de particionamiento de datos pooled. Las economías de escala y el perfil operacional se benefician ambos del uso de infraestructura compartida. Sin embargo, el dimensionamiento del dominio puede ser difícil y puede llevar al sobredimensionamiento.

Este es otro escenario donde apoyarse en tecnologías serverless puede aportar valor real y orientar hacia el servicio OpenSearch Serverless, que puede simplificar el perfil de dimensionamiento y escalado de la solución.

Particionamiento de OpenSearch en modelo siloed

Hay dos opciones para implementar un modelo de particionamiento de datos siloed en OpenSearch. La primera opción es el modelo de dominio por tenant. Con este enfoque, será necesario crear un cluster completamente separado para cada tenant.

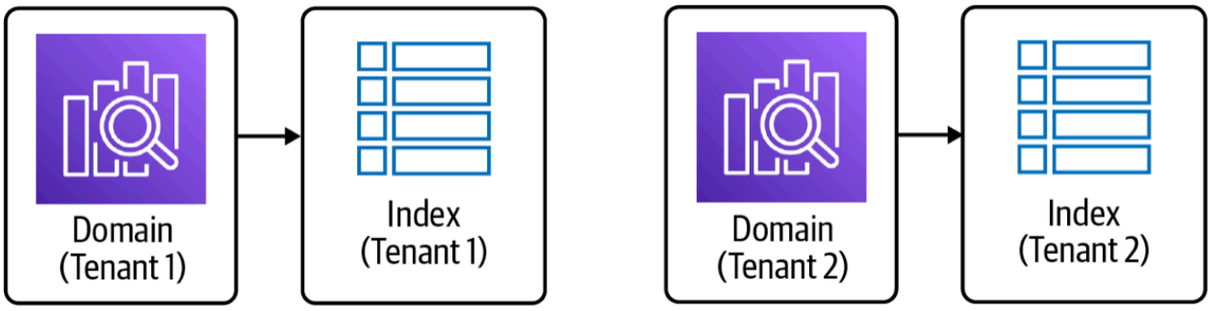

Figure 8-12. Domain-per-tenant OpenSearch siloed data partitioning

Al mismo tiempo, esto añadirá complejidad operacional e impactará la eficiencia de costos del entorno.

La otra forma de implementar el particionamiento de datos siloed con OpenSearch es a través de un modelo de índice por tenant.

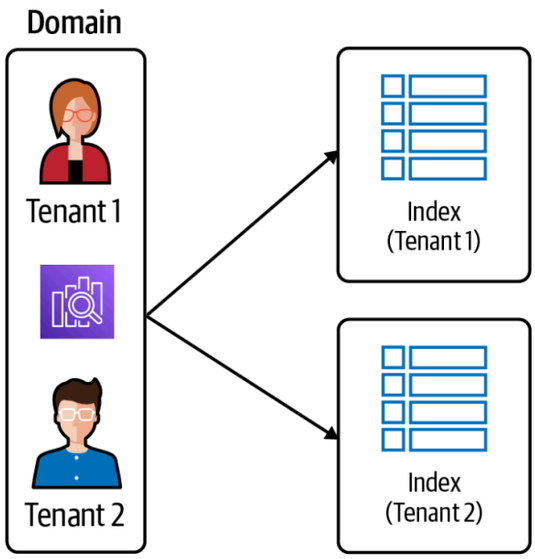

Figure 8-13. Index-per-tenant siloed OpenSearch data partitioning

En la parte superior se muestran dos tenants del tier premium con índices dedicados. En la parte inferior, hay un índice pooled que combina todos los datos del tier básico en un único elemento.

Fragmentación de datos de tenants (Sharding)

Hay momentos en que la escalabilidad y el rendimiento del modelo de almacenamiento no pueden satisfacer las necesidades de la solución. En estos casos, puede ser necesario considerar la introducción de mecanismos propios para fragmentar el almacenamiento de tenants. **La fragmentación generalmente se refiere a la idea de dividir un recurso en múltiples partes para abordar la escalabilidad, el tamaño, y otros aspectos.**

Un patrón que se ha aplicado es el del particionamiento personalizado, donde el código asume la responsabilidad de mapear tenants a diferentes elementos de almacenamiento.

Sin embargo, se descubre que la escalabilidad y el rendimiento de los tenants superan la capacidad de atender adecuadamente su carga de trabajo. Se prefiere mantener los tenants en modo pooled, pero resulta poco práctico tenerlos todos en una sola base de datos de almacenamiento.

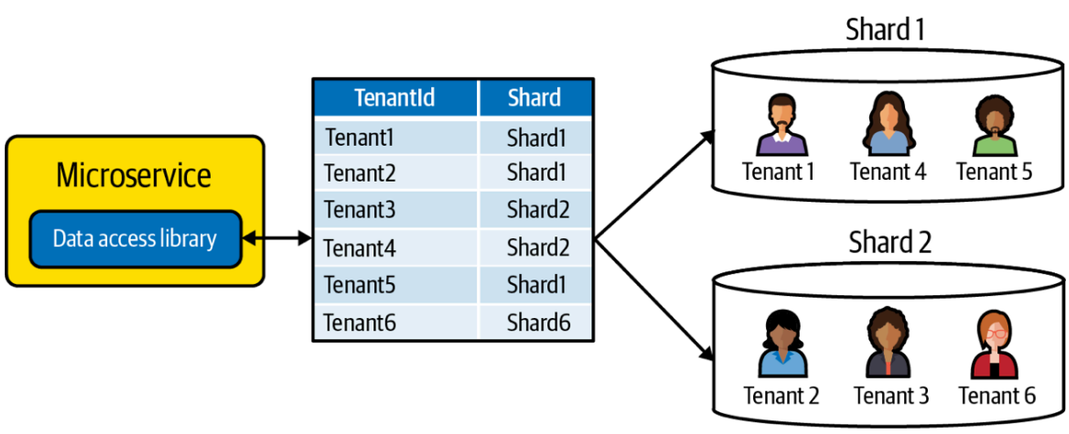

Figure 8-15. Sharding groups of tenants

Esto es muy similar al modelo de despliegue en pods que se analizó en el Capítulo 3, donde grupos de tenants se desplegaban juntos en pods. Aquí se aplica un concepto similar, pero exclusivamente en la capa de almacenamiento del entorno.

Este modelo está lejos de ser ideal y sin duda añade complejidad y fricción al entorno SaaS. Sin embargo, se incluye aquí porque algunas cargas de trabajo o requisitos de negocio pueden considerarlo un compromiso razonable.

Consideraciones sobre el ciclo de vida de los datos

Hay diferentes cambios de estado que un tenant puede atravesar y que podrían impactar la forma en que sus datos están representados.

Cambio de tier

Los tenants del tier básico utilizan datos completamente pooled, mientras que los tenants del tier premium tienen algunos elementos de almacenamiento siloed.

Imaginemos qué significará tener un tenant que necesita actualizar del tier básico al tier premium. En este caso, sería necesario contar con automatización que pudiera migrar los datos de forma fluida desde el entorno pooled a un modelo siloed.

La automatización también debería considerar cómo el traslado de estos datos podría impactar la carga general del sistema.

Este es un problema difícil para el que existen pocas soluciones elegantes. Aun así, es algo que debe estar en el radar.

Baja del tenant

La mayor parte del proceso de baja de un tenant suele centrarse en qué se planea hacer con sus datos.

- ¿Se recorrerá el sistema completo para eliminar los datos del tenant?
- ¿Se archivarán los datos, permitiendo al tenant restaurarlos cuando regrese?

Esta es otra área donde será necesario construir cuidadosamente las herramientas de automatización que puedan ejecutar las políticas de baja, aplicándolas sin degradar el rendimiento del entorno.

Respaldo y restauración

Esto puede ser especialmente complicado cuando se tienen datos pooled.

Además, es probable que los datos de los tenants estén distribuidos en múltiples elementos de almacenamiento del entorno, utilizando potencialmente una variedad de modelos siloed y pooled. Esto, como puede imaginarse, requiere un mecanismo cuidadosamente orquestado que pueda adquirir y respaldar los datos de los tenants de forma exitosa.

Para algunos, no existe el concepto de respaldo y restauración por tenant. En cambio, el estado de los datos se considera un elemento global.

Seguridad de datos multi-tenant

La protección de los datos de los tenants es un requisito básico de cualquier entorno SaaS, y la arquitectura ya debería contemplar medidas sólidas para garantizar que un tenant no pueda acceder a los recursos de otro.

Sin embargo, hay dominios que pueden imponer requisitos de seguridad adicionales sobre el almacenamiento de sus datos. Para algunos, esto puede estar más relacionado con cifrar los datos para garantizar su protección en reposo y en tránsito.

La capacidad de cifrar los datos está en gran medida determinada por las capacidades de cifrado de los servicios de almacenamiento utilizados.

Para algunos, el cifrado de los datos puede seguir siendo insuficiente. Puede haber tenants que requieran cifrado y titularidad sobre las claves utilizadas para gestionar el acceso a sus datos. En estos escenarios, será necesario introducir elementos para crear y entregar estas claves a tenants individuales. **Estas claves también pueden tener un ciclo de vida que deberá gestionarse como parte de la huella de recursos operacional del entorno SaaS.**