

Capítulo 3

Computación In-Memory

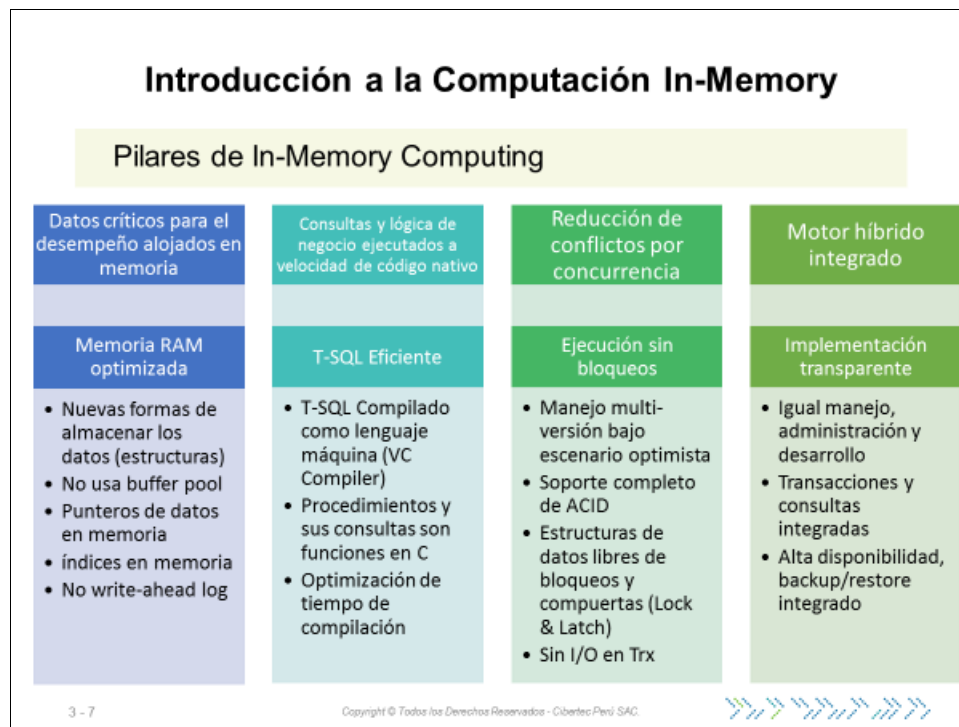
Al finalizar el capítulo, el alumno podrá:

- Conocer las características y beneficios de la computación In-Memory.
- Implementar tablas e índices optimizados para computación In-Memory.
- Implementar código optimizado para computación In-Memory.
- Administrar bases de datos optimizadas para computación In-Memory.

Temas

1. Introducción a la computación In-Memory
2. Tablas optimizadas para uso In-Memory
3. Índices optimizados para uso In-Memory
4. Código optimizado para uso In-Memory
5. Administración de bases de datos con objetos optimizados para In-Memory

1. Introducción a la computación In-Memory



1.1 Características y beneficios de computación In-Memory

Implementar computación In-Memory mejora el rendimiento de escenarios de datos temporales, carga de datos, y procesamiento de transacciones.

La computación In-Memory proporciona ganancias de rendimiento para las cargas de trabajo adecuadas. A pesar de que ciertos clientes han visto una ganancia de rendimiento de hasta 30 veces en algunos casos, la ganancia que obtendrá depende de la carga de trabajo.

La ganancia de rendimiento proviene del procesamiento de transacciones al hacer que la ejecución de las transacciones y el acceso a los datos sea más eficaz, al quitar la contención de bloqueo y bloqueo temporal entre transacciones que se ejecutan de manera simultánea

Es importante entender que no es rápido porque se haga en memoria, sino que es rápido porque está optimizado en torno a los datos que están en memoria. Los algoritmos de procesamiento, acceso y almacenamiento de datos se rediseñaron desde el principio para aprovechar las mejoras más recientes en los cálculos de alta simultaneidad y en memoria.

Solo porque los datos residen en memoria, no significa que los pierda si se produce un error. De manera predeterminada, todas las transacciones son duraderas, lo que significa que tiene las mismas garantías de durabilidad que obtiene para cualquier otra tabla en SQL Server. Como parte de la confirmación de transacciones, todos los cambios se escriben en el registro de transacciones en el disco. Si se produce algún error en cualquier momento después de la confirmación de la transacción, los datos se mantienen cuando la base de datos vuelve a estar en línea.

Para usar computación In-Memory, se debe implementar uno o varios de los siguientes tipos de objetos:

- Tablas con optimización para memoria
- Tablas no duraderas
- Módulos T-SQL compilados de manera nativa

2. Escenarios adecuados para el uso de computación In-Memory

Escenarios adecuados para el uso de computación In-Memory

In Memory OLTP con SQL Server 2014

- Primer release de In-Memory Computing
- Aumento extremo del rendimiento de transacciones (30 veces mas rápido)
- Alta escalabilidad: arquitectura libre de lock y latch, escalado lineal:
Acceso y procesamiento de datos optimizado: Estructura de datos en memoria y compilación nativa

- Integrado a plataforma SQL Server
- Edición Enterprise
- Implementación híbrida In-Memory / On-Disk
- Desarrollo integrado (T-SQL, backup/restore, Always On)

Escalado tradicional

Rendimiento/sec

Número de hilos

Escalado lineal

Rendimiento/sec

Número de hilos

Escenario objetivo: aplicaciones OLTP con alto volumen de procesamiento y crecimiento de datos

3 - 8

Copyright © Todos los Derechos Reservados - Cibertec Perú S.A.C.

2.1 Procesamiento de transacciones de baja latencia y alto rendimiento

Este es el escenario central para el que compilamos OLTP en memoria: se admiten grandes volúmenes de transacciones con una baja latencia coherente para transacciones individuales.

Los escenarios de carga de trabajo comunes son: comercialización de instrumentos financieros, apuestas deportivas, juegos móviles y publicación de anuncios. Otro patrón común que hemos visto es un "catálogo" que se lee o actualiza con frecuencia. Un ejemplo es donde tiene archivos de gran tamaño, cada uno de ellos distribuido sobre un número de nodos en un clúster, y se cataloga la ubicación de cada partición de cada archivo en una tabla con optimización para memoria.

2.2 Big Data, incluida IoT (Internet de las cosas)

Computación In-Memory es excelente en el procesamiento de grandes volúmenes de datos desde distintos orígenes al mismo tiempo. Además, la ingesta de datos en una base de datos de SQL Server suele ser beneficiosa en comparación con otros destinos, porque SQL permite ejecutar realmente rápido las consultas de datos y, además, le permite obtener información en tiempo real.

Los patrones de aplicaciones comunes son: la ingestión de eventos y lecturas de sensores, para permitir la notificación, además del análisis histórico. La

administración de actualizaciones por lotes, incluso desde varios orígenes, mientras se minimiza el impacto en la carga de trabajo de lectura simultánea.

ETL (extracción, transformación, carga)

A menudo, los flujos de trabajo de ETL incluyen cargar datos en una tabla provisional, transformaciones de los datos y su carga en las tablas finales

3. Tablas e índices optimizados para uso In-Memory.

Implementar tablas e índices optimizados para computación In-Memory

```
CREATE TABLE dbo.MiTabla_MOD (  
    Mtmod_ID int IDENTITY(1,1) NOT NULL,  
    Mtmod_Fecha datetime NOT NULL,  
    Mtmod_Status int NOT NULL  
  
    CONSTRAINT PK_MiTabla_MOD PRIMARY KEY  
        NONCLUSTERED HASH (Mtmod_ID) WITH (BUCKET_COUNT=20)  
) WITH (MEMORY_OPTIMIZED=ON, DURABILITY=SCHEMA_AND_DATA)  
GO
```

3 - 12

Copyright © Todos los Derechos Reservados - Cibertec Perú S.A.C.



A partir de SQL Server 2016, no hay límite para el tamaño de las tablas con optimización para memoria más que la memoria disponible. En SQL Server 2014, el tamaño total en memoria de todas las tablas durables de una base de datos no debería superar los 250 GB para las bases de datos de SQL Server 2014.

Las tablas con optimización para memoria son tablas creadas por medio de CREATE TABLE (Transact-SQL).

De forma predeterminada, las tablas con optimización para memoria son totalmente durables y, al igual que las transacciones en tablas basadas en disco (tradicionales), las transacciones totalmente durables en este tipo de tablas tienen todas las propiedades de atomicidad, coherencia, aislamiento y durabilidad (ACID). Las tablas con optimización para memoria y los procedimientos almacenados compilados de forma nativa admiten un subconjunto de Transact-SQL.

A partir de SQL Server 2016, y en Azure SQL Database, no existen limitaciones para intercalaciones o páginas de códigos que son específicas de OLTP en memoria.

El almacén principal para las tablas con optimización para memoria es memoria principal; las tablas con optimización para memoria residen en memoria. Las filas de la tabla se leen y se escriben en la memoria. Toda la tabla reside en memoria. Una segunda copia de los datos de la tabla se conserva en el disco pero solo por la durabilidad. Vea Crear y administrar el almacenamiento de objetos con optimización para memoria para obtener más información sobre las tablas durables. Los datos de las tablas con optimización para memoria solo se leen del disco durante la recuperación de la base de datos. Por ejemplo, después de reiniciar el servidor.

Se puede tener acceso a las tablas con optimización para memoria de forma más eficaz desde procedimientos almacenados compilados de forma nativa (Procedimientos almacenados compilados de forma nativa). Se puede tener acceso también a las tablas con optimización para memoria con Transact-SQL interpretado (tradicional). Transact-SQL interpretado hace referencia al acceso a tablas con optimización para memoria sin un procedimiento almacenado compilado de forma nativa. Algunos ejemplos de acceso con Transact-SQL interpretado son el acceso a una tabla con optimización para memoria desde un desencadenador DML, un lote ad hoc de Transact-SQL, una vista y una función con valores de tabla

4. Código optimizado para uso In-Memory


Implementar código optimizado para computación In-Memory

```
CREATE PROCEDURE dbo.InsertaMtmod
WITH NATIVE_COMPILATION, SCHEMABINDING
AS
BEGIN ATOMIC WITH (TRANSACTION ISOLATION LEVEL = SNAPSHOT,
LANGUAGE='spanish')
DECLARE
    @ID int = 1,
    @Status tinyint = 1,
    @Fecha datetime = getdate()

WHILE @ID <= 100000
BEGIN
    INSERT INTO dbo.MiTabla_MOD (Mtmod_Fecha,Mtmod_Status)
    VALUES (@Fecha, CAST(RAND()*1000 AS int))
    SET @ID += 1
END
END
GO
```

3 - 15

Copyright © Todos los Derechos Reservados - Cibertec Perú S.A.C.



Los procedimientos almacenados compilados de forma nativa son los procedimientos almacenados de Transact-SQL compilados para código nativo que tienen acceso a las tablas con optimización para memoria. Los procedimientos almacenados compilados de forma nativa permiten la ejecución eficaz de consultas y la lógica empresarial en el procedimiento almacenado

Una diferencia entre los procedimientos almacenados (basados en disco) interpretados y los procedimientos almacenados compilados de forma nativa es que un procedimiento almacenado interpretado se compila en la primera ejecución mientras que un procedimiento almacenado compilado se compila cuando se crea. Con los procedimientos almacenados compilados de forma nativa, muchas condiciones de error se pueden detectar en el momento de la creación y harán que la creación del procedimiento almacenado compilado de forma nativa genere un error (desbordamiento aritmético, conversión de tipo y otras condiciones de división por cero). Con los procedimientos almacenados interpretados, estas condiciones de error normalmente no provocan un error al crear el procedimiento almacenado, pero todas las ejecuciones producirán un error.

En el ejemplo de código, NATIVE_COMPILATION indica que este procedimiento almacenado de Transact-SQL es un procedimiento almacenado compilado de forma nativa. Se requieren las siguientes opciones:

Opción	Descripción
SCHEMABINDING	<p>Un procedimiento almacenado compilado de forma nativa se debe enlazar al esquema de objetos al que hace referencia. Esto significa que no se puede anular las tablas a las que hace referencia el procedimiento. Las tablas a las que se hace referencia en el procedimiento deben incluir el nombre de esquema y no se admiten caracteres comodín (*) en las consultas (es decir, sin <code>SELECT * from...</code>). SCHEMABINDING solo se admite para los procedimientos almacenados compilados de forma nativa en esta versión de SQL Server.</p>
BEGIN ATOMIC	<p>El cuerpo de un procedimiento almacenado compilado de forma nativa debe constar exactamente de un solo bloque atomic. Los bloques atomic garantizan la ejecución atómica del procedimiento almacenado. Si se invoca el procedimiento fuera del contexto de una transacción activa, iniciará una nueva transacción, que se confirma al final del bloque atomic. Los bloques atomic de los procedimientos almacenados compilados de forma nativa tienen dos opciones obligatorias:</p> <p>TRANSACTION ISOLATION LEVEL. Niveles de aislamiento de transacción para tablas con optimización para memoria</p> <p>LANGUAGE. El lenguaje del procedimiento almacenado se debe establecer en uno de los lenguajes o de alias de lenguaje disponibles.</p>

BEGIN ATOMIC es parte del estándar ANSI SQL. SQL Server admite bloques ATOMIC en el nivel superior de los procedimientos almacenados compilados de forma nativa, así como en las funciones escalares definidas por el usuario y compiladas de forma nativa. Para obtener más información sobre estas funciones, vea Funciones escalares definidas por el usuario para OLTP en memoria.

- Cada procedimiento almacenado compilado de forma nativa contiene exactamente un bloque de instrucciones Transact-SQL. Este es un bloque ATOMIC.
- Los procedimientos almacenados Transact-SQL interpretados no nativos y los lotes ad hoc no admiten los bloques atomic.

Los bloques atomic se ejecutan (atómicamente) dentro de la transacción. Todas las instrucciones del bloque se ejecutan correctamente o el bloque completo se revertirá al punto de retorno que se creó al principio del bloque. Además, los parámetros de configuración de la sesión son fijos para el bloque atomic. La ejecución del mismo bloque atomic en sesiones con diferentes parámetros producirá el mismo comportamiento, independientemente de la configuración de la sesión actual.