SQL Server 2019: In-Memory Tabellen

Kathegorie
SQL Server

Aaron
Priesterroth

Einer der häufigsten Gründe für Latenzen in der Benutzung oder Entwicklung von Anwendungen ist der Zugriff auf Anwendungsdaten.
Die beiden kostspieligsten Aktionen, die innerhalb der Anwendungsentwicklung durchgeführt werden können, sind das Lesen und Schreiben von Informationen auf die Festplatte sowie der Aufbau einer Verbindung zu einem anderen Computer. Mit einem Zugriff auf einen Datenbank-Server werden diese beiden Aktionen sogar gleichzeitig ausgeführt.

Abhilfe kann dabei vor allem das Konzept des **Cachens** leisten: werden die Daten vom Benutzer lokal gespeichert, müssen keine Abfragen getätigt werden. In der Praxis gelingt dies jedoch nur bei sehr statischen Daten. Werden die verwendeten Daten von vielen Anwendern gleichzeitig benötigt und verändert, liefert **Cachen** nur bedingt eine Lösung.

Server-seitiges Cachen mit Speicher-optimierten Tabellen

Eine bessere Lösung ist es, das Caching an der Stelle des Datenbank-Servers durchzuführen. Mit der ersten Abfrage eines Benutzers werden die benötigten Informationen von der Festplatte geladen und bereitgestellt. Werden diese Informationen nun im Speicher behalten, können diese von allen folgenden Abfragen ebenfalls direkt verwendet werden.

An dieser Stelle kommen die in SQL Server 2014 eingeführten In-Memory OLTP Tabellen ins Spiel: auf Festplatten-Zugriffe wird hier komplett verzichtet. Stattdessen wird die Tabelle im Speicher gehalten. Der Umgang mit Daten-Zugriffen erfährt dadurch eine Art Renaissance, vor allem die Performanz wird in der Regel erheblich verbessert.

Die Tatsache, dass die Tabelle im RAM verwaltet wird, lässt vermuten, dass die in der Tabelle befindlichen Daten verloren gehen, sobald es zu einem Ausfall des Servers kommt. Tatsächlich aber wird eine Kopie jeder Transaktion im Transaktions-Log gespeichert. Kommt es also zum Ausfall eines Servers, kann dieser nach erfolgreichem Neustart die Tabelle anhand des Logs neu aufbauen und den Zustand vor dem Ausfall wiederherstellen.

Es besteht sogar die Möglichkeit, eine Temporale Tabelle als Speicher-optimierte Tabelle zu realisieren. In diesem Szenario wird die Historie der Tabelle auf der Festplatte gespeichert, während die aktiven Daten im Speicher verwaltet werden. Dabei verwaltet der SQL Server selbstständig die Verschiebung veralteter Informationen auf die Festplatte, um immer genügend RAM für neue Informationen zu bieten.

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

Limitationen

Als die Speicher-optimierten Tabellen mit SQL Server 2014 veröffentlicht wurden, gab es eine ganze Reihe von Limitationen:

- Die Struktur einer Tabelle konnte nachträglich nicht mehr verändert werden.
- Viele SQL Operatoren (OUTER JOIN, Subqueries in SELECT, DISTINCT, OR, NOT, etc.) wurden nicht unterstützt.
- Lediglich die "Latin" Collation stand zur Verfügung.

Diese Limitationen wurden mit der Veröffentlichung von SQL Server 2016 behoben.

Dennoch existieren nach wie vor eine Reihe von Limitationen, die vor der Verwendung dieser Konzept bedacht werden sollten:

- Daten-Kompression kann nicht verwendet werden.
- Replikation kann nur zwischen In-Memory Tabellen stattfinden.
- Server- und Datenbank-Ebenen Trigger werden nicht unterstützt.

Zusätzlich muss die Größe der Tabelle berücksichtigt werden. In vielen Fällen sind Tabellen zu groß, um vollständig im Speicher gehalten zu werden. Eine Möglichkeit ist es, nach Paretos 80/20 Regel zu agieren: wenn beispielsweise 80 Prozent eines Unternehmens von 20 der Produkte abhängen, empfehlen wir nur die wichtigsten Produkte im RAM zu speichern und die weniger wichtigen Produkte auf der Festplatte zu lagern. Eine Variation dieser Herangehensweise mit In-Memory Tabellen ist es, lediglich Schlüssel-Spalten an Stelle der Schlüssel-Zeilen im Speicher zu behalten. Ein In-Memory Table besteht dann beispielsweise nur noch aus der Produkt-ID und den jeweiligen verfügbaren Einheiten des Produkts.

Indexing

Für In-Memory Tabellen Indexe stehen Hash-Index und Non-Clustered-Index zur Verfügung. Clustered-Indexe werden nicht unterstützt.
Für Point-Queries, also Abfragen, die auf ein einzelnes Ergebnis abzielen, sollte dabei auf den Hash-Index zurückgegriffen werden. Für Range-Queries, also Abfragen, die auf einen Bereich von Werten abzielen, sollte statt dessen ein Non-Clustered_index verwendet werden.

Für beide Index-Typen gilt: eine hohe Anzahl an Duplikaten (mehr als 100 Duplikat-Zeilen für gleiche Werte) wirken sich negativ auf die Performanz aus. In einem solchen Falle empfiehlt Microsoft die Verwendung eines Non-Clustered_index und der zusätzlichen Erweiterung des Index durch eine (oder mehrere) weitere Spalte(n).

In-Memory Tabellen erstellen

Um eine In-Memory Tabelle erstellen zu können, muss eine Datei-Gruppe in der Datenbank existieren. Es darf dabei nur eine solche Datei-Gruppe existieren.

```
ALTER DATABASE my_database
  ADD FileGroup my_file_group
  CONTAINS MEMORY_OPTIMIZED_DATA;

ALTER DATABASE my_database
  ADD FILE (name='my_file_group_con1', filename='C:\LocalData\my_file_
group_con1')
  TO FILEGROUP my_file_group;
```

Danach kann die In-Memory Tabelle auch schon erstellt werden: mit der **WITH**-Klausel wird angegeben, dass die zu erstellende Tabelle als In-Memory Tabelle erstellt werden soll:

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

```
CREATE TABLE [dbo].[Test] (
  f_name NVARCHAR(1000) NOT NULL,
  l_name NVARCHAR(1000) NOT NULL,
  age INT NOT NULL)
  WITH (Memory_Optimized = ON);
```

Standardmäßig wird durch die Option Memory_Optimized = ON eine ”haltbare“ Tabelle erzeugt. Falls der Verlust der Daten bei einem Herunterfahren des Server irrelevant ist, gibt es zusätzlich die Möglichkeit, eine Haltbarkeitsoption anzugeben: SCHEMA_ONLY.

```
CREATE TABLE [dbo].[Test] (
  f_name NVARCHAR(1000) NOT NULL,
  l_name NVARCHAR(1000) NOT NULL,
  age INT NOT NULL)
WITH (Memory_Optimized = ON, DURABILITY = SCHEMA_ONLY);
```


Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612