Kapitel 2
SAP HANA als Datenbank

Den Kern eines SAP-HANA-Systems bildet die Datenbank, deren Struktur wir Ihnen in diesem Kapitel vorstellen. SAP HANA unterstützt unterschiedliche Strategien zur Datenhaltung: im Speicher und auf der Festplatte. Zusätzlich stellt SAP HANA Kommunikationsschnittstellen bereit, um mit externen Systemen kommunizieren zu können. Sowohl die Datenhaltung als auch die Kommunikation werden in diesem Kapitel besprochen.

Wir nähern uns SAP HANA als Datenbank in diesem Kapitel in fünf Abschnitten. In Abschnitt 2.1 stellen wir Ihnen das ACID-Prinzip für Datenbanktransaktionen vor und beschreiben, wie dieses Prinzip in SAP HANA umgesetzt wurde. In Abschnitt 2.2, »Tabellenstrukturen im Speicher von SAP HANA«, gehen wir genauer auf die Tabellenstrukturen ein. SAP HANA setzt generell auf reihenbasierte Tabellenstrukturen mit einem zusätzlichem Deltaspeicher. In Abschnitt 2.3, »Kommmunikationsschnittstellen«, stellen wir Ihnen die verschiedenen Schnittstellentechnologien vor, die von der SAP-HANA-Datenbank verwendet werden.

Eine Besonderheit von SAP HANA als Datenbank ist, dass sie die Datenhaltung auch außerhalb des Arbeitsspeichers umsetzen kann. Welche Daten wo persistiert werden, ist auch von der Art der Daten abhängig. Die drei zur Verfügung stehenden Möglichkeiten für die Datenhaltung stellen wir Ihnen in Abschnitt 2.4, »Warm und Hot Storage«, vor. In Abschnitt 2.5 erklären wir abschließend noch, was Tabellenpartitionierung ist und wie diese in SAP HANA umgesetzt werden kann.

2.1 Das ACID-Prinzip

Wie bereits erwähnt, ist SAP HANA eine In-Memory-Datenbank. Die Datensätze befinden sich im Normalfall zu 100 % im Arbeitsspeicher der Maschine. Es gibt jedoch auch die Möglichkeit, Daten außerhalb des Arbeitsspeichers zu halten, wie es bei traditionellen Datenbanken der Fall ist. Dieser Möglichkeit bedient man sich z. B. bei weniger häufig abgefragten Daten (siehe Abschnitt 2.4, »Warm und Hot Storage«). Doch selbst,

wenn man diese Datenbank-Features intensiv nutzt, befinden sich immer noch sehr viele Datensätze ausschließlich im Arbeitsspeicher.

Anforderungen der In-Memory-Datenhaltung

Dieser Umstand stellt besondere Anforderungen an die Durchführung von Datenbanktransaktionen auf der SAP HANA Engine. Denn wenn die Host-Maschine ausfällt, darf es trotz des Ausfalls nicht zu Datenverlust kommen. Eine Datenbank, die bei einem Ausfall des Stroms, der Hardware oder durch einen Softwarefehler Datensätze verliert, wäre für einen produktiven Einsatz in Unternehmen ungeeignet, denn niemand würde es riskieren, operative Daten zu verlieren. Deshalb muss eine moderne Datenbank wie SAP HANA in der Lage sein, den letzten Zustand vor einem Ausfall jederzeit zu rekonstruieren. Wäre sie dies nicht und ließen sich die verloren gegangenen Datensätze auch in den Datenquellen nicht mehr ausmachen, träte der Worst Case ein.

ACID-Prinzip

SAP setzt deshalb bei der SAP-HANA-Datenbank auf das *ACID-Prinzip* für Datenbanktransaktionen. ACID steht für Atomicity, Consistency, Isolation und Durability, zu Deutsch *Atomarität*, *Konsistenz*, *Isolation* und *Dauerhaftigkeit*. Jeder dieser vier Teilaspekte des ACID-Prinzips ist wichtig für einen möglichst verlustfreien Datenbankbetrieb. Grund genug, sich diese vier Aspekte in diesem Abschnitt einmal näher anzusehen.

Atomarität

Atomarität bedeutet, dass alle einzelnen Instruktionen einer Transaktion als Einheit zu sehen sind. Eine Transaktion kann aus verschiedenen SQL-Anweisungen wie `SELECT`, `UPDATE` etc. bestehen. Sollte eine Anweisung innerhalb einer Transaktion fehlschlagen, gilt die gesamte Transaktion als fehlgeschlagen und nicht nur die fehlerhafte Anweisung. Deshalb kommt es im Fehlerfall zum *Rollback* der gesamten Transaktion, d. h. zur Wiederherstellung des letzten konsistenten Zustands vor der Ausführung der Transaktion. Aufgrund dieses Verhaltens wird die Umsetzung des Atomaritätsprinzips auch als *All-or-nothing-Vorgehen* bezeichnet.

Konsistenz

Unter einem konsistenten Zustand im Kontext von Datenbanken versteht man, dass alle Daten korrekt sind. Datenkonsistenz ist wichtig für den Betrieb einer Datenbank, da auf der Applikationsschicht (*Application Layer*) einer betriebswirtschaftlichen Software implizit davon ausgegangen wird, dass die Datensätze konsistent (richtig) sind. Die Anwendungsschicht ist die Schicht, auf der die Softwareprogramme ausgeführt werden. In SAP HANA gehören z. B. die SAP HANA Extended Application Services (XS Engine) oder die Extended Application Services, Advanced Model (XSA Engine), zur Anwendungsschicht. Auf die SAP HANA XS Engine und die SAP XSA Engine gehen wir in Abschnitt 8.2.1, »SAP HANA Extended Application Services, Classic Model«, und Abschnitt 8.2.2, »SAP HANA Extended Application Services, Advanced Model«, näher ein. Aber auch auf der Ebene der

betriebswirtschaftlichen Anwendungen in SAP S/4HANA müssen die Daten konsistent vorliegen. Inkonsistenzen im Datenbestand können folglich zu schwerwiegenden Fehlern auf der Anwendungsschicht führen. Sollte durch Ausführen einer Transaktion kein konsistenter Zustand hergestellt werden können, wird die gesamte Transaktion daher zurückgespielt.

Konsistenzchecks

Während des Betriebs führt SAP HANA dazu automatisch Konsistenzchecks durch. Sie können jedoch auch manuelle Konsistenzchecks ausführen lassen, sollten Sie dies für notwendig halten. Werden Inkonsistenzen bei einem Check festgestellt, werden diese protokolliert. Die Protokolle können Sie über die View `GLOBAL_TABLE_CONSISTENCY` einsehen. Dazu verwenden Sie am besten folgende SQL-Anweisung:

```
SELECT TOP * FROM "_SYS_STATISTICS"."GLOBAL_TABLE_CONSISTENCY"
WHERE "SEVERITY" != 'INFO'
```

Sie können SQL-Anweisungen wie diese in verschiedenen Administrationswerkzeugen ausführen. Mehr dazu erfahren Sie in Kapitel 6, »Administrationsumgebungen«.

Sie sollten bei Abfragen auf diese View, wie hier gezeigt, mit einer `WHERE`-Klausel den Wert `'INFO'` in der Tabellenspalte `SEVERITY` herausfiltern. Damit erhalten Sie als Rückgabewert nur tatsächliche Inkonsistenzen, da erfolgreiche Konsistenzprüfungen in der Tabelle mit dem Wert `'INFO'` gekennzeichnet werden. Einen tieferen Einblick in das Thema Konsistenzchecks erhalten Sie in Abschnitt 5.6.2.

Isolation

Sehen wir uns als Nächstes das Prinzip der Isolation an. In SAP HANA laufen Transaktionen häufig parallel, um die zur Verfügung stehenden Rechenkerne auszulasten. Jede Transaktion, egal ob parallel oder seriell ausgeführt, muss immer dasselbe Ergebnis liefern. Greifen Transaktionen auf unterschiedliche Tabellen zu, kommt es nicht zu Überschneidungen, und die Isolationsregeln werden automatisch eingehalten. Greifen jedoch mehrere Transaktionen auf denselben Datenbestand zu, müssen diese Zugriffe geregelt werden, um die Isolationsregel nicht zu verletzen. Dies wird erreicht, indem eine Transaktion bei einem Tabellenzugriff diese Tabelle für andere Transaktionen sperrt. Diese Sperre wird erst wieder aufgehoben, wenn die sperrende Transaktion mit ihrem Zugriff auf die Tabelle fertig ist.

In der SAP-Welt ist das Sperren von Tabellen ein großes Thema, da sehr viele Transaktionen auf sehr viele Tabellen zugreifen. Es gibt unterschiedliche Strategien, um Objekte zu sperren. Sollten Sie sich näher mit diesem Themenkomplex im Kontext von SAP HANA befassen wollen, ist der SAP-Hinweis 1999998 eine gute Quelle. Wir werden im Verlauf dieses Buches noch häufiger auf solche *SAP-Hinweise* verweisen. In Abschnitt 12.4.3, »SAP

Support Portal«, erklären wir Ihnen, wie Sie die SAP-Hinweise aufrufen können. In SAP-Hinweis 1999998 werden die unterschiedlichen Sperrverfahren vorgestellt. Ergänzend werden die Views aufgelistet, mit denen Sie Einblick gewinnen können, welche Objekte gerade mit welcher Methode und von welchem Prozess gesperrt sind.

Dauerhaftigkeit

Der vierte Aspekt des ACID-Prinzips, die Dauerhaftigkeit, bedeutet im Datenbankumfeld, dass das Ergebnis einer erfolgreich (also ohne das Auftreten von Inkonsistenzen) durchgeführten Transaktion permanent gespeichert sein muss. Der neue konsistente Zustand muss also auch nach einem Datenbankausfall noch verfügbar sein.

Persistenzschicht

Zur Sicherstellung der Dauerhaftigkeit dient in SAP HANA die Persistenzschicht (*Persistance Layer*). In Abbildung 2.1 ist die Architektur der SAP-HANA-Datenbank dargestellt. Die Persistenzschicht ist hier im Index Server verortet. Näheres zum Index Server erfahren Sie in Abschnitt 3.1.1. Die Persistenzschicht regelt die Kommunikation zwischen den Datenbank-Engines und dem Storage der Datenbank. Er sorgt so dafür, dass SAP HANA ACID-konform arbeiten kann.

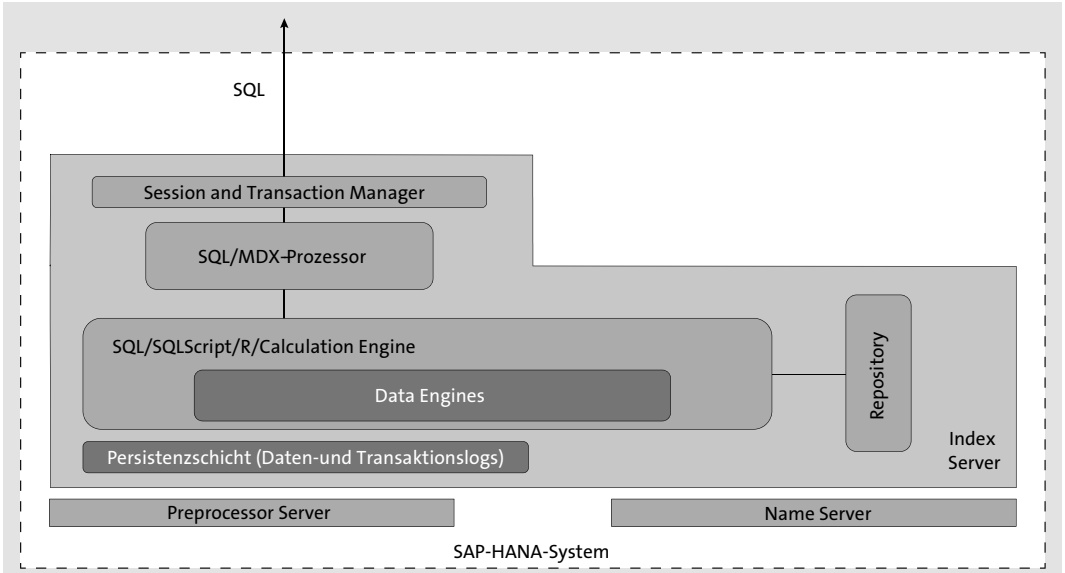

Abbildung 2.1 Datenbankarchitektur von SAP HANA (Quelle: SAP)

Speicherbereiche

Um sicherzustellen, dass bei einem Datenbankausfall kein Datenverlust auftritt, verwendet die Persistenzschicht zwei Speicherbereiche. Diese heißen **data** und **log**. Sie befinden sich auf dem Storage der Datenbank. Die

Standardpfade für den Zugriff auf diese Speicherbereiche sind **/hana/log** und **/hana/data**.

- **Speicherbereich »data«**
  In der Standardkonfiguration schreibt SAP HANA alle 300 Sekunden (also alle fünf Minuten) alle Datenbankänderungen in einen Speicherpunkt im **data**-Bereich. Die zu speichernden Änderungen haben meist eine Größenordnung von 4 KB bis 16 MB. Der Schreibvorgang erfolgt dabei asynchron.

  Der **data**-Bereich allein reicht bei einem Ausfall jedoch nicht aus, um eine Dauerhaftigkeit im Sinne des ACID-Prinzips zu gewährleisten. Fiele er während des asynchronen Schreibens oder zu einem beliebigen Zeitpunkt innerhalb des 5-Minuten-Zyklus aus, käme es daher zu einem Datenverlust.

- **Speicherbereich »log«**
  Um diesen Zeitraum zu überbrücken, wird daher zusätzlich der Speicherbereich **log** eingesetzt. Darin werden alle Schreibvorgänge immer synchron ausgeführt. Im **log**-Bereich werden alle ausgeführten Transaktionen gespeichert. Eine Transaktion gilt in SAP HANA immer erst dann als ausgeführt, wenn sie in den **log**-Bereich geschrieben wurde. Eine Ausführung der Transaktion durch die Datenbank-Engine allein reicht dafür nicht aus. Durch diese Strategie ist jede ausgeführte Transaktion auch immer rekonstruierbar, wie von ACID gefordert.

  Abhängig von der Frequenz der Sicherungen in den **data**-Bereich, der Anzahl der Transaktionen und der eingestellten Log-Methode kann es vorkommen, dass der **log**-Bereich sehr groß wird. Ist dieser Speicherbereich voll, ist die Datenbank nicht mehr funktionsfähig, denn es können keine Transaktionen mehr persistiert werden, was weitere Ausführungen durch die Datenbank-Engine unmöglich macht. Sollte dies der Fall sein, können Sie versuchen, den **log**-Bereich aufzuräumen. Dazu führen Sie folgende SQL-Anweisung aus:

```
ALTER SYSTEM RECLAIM LOG;
```

  Sollte dies nicht ausreichen oder die SQL-Anweisung sogar ohne Wirkung bleiben, bleibt Ihnen nur eine Vergrößerung des Speicherbereichs übrig. Den optimalen Aufbau des Dateisystems, um in solch einer kritischen Situation schnell reagieren zu können, erläutern wir in Abschnitt 5.2.1, »Vorbereitung der Installation«.

Performance des »log«-Bereichs

Sie sollten außerdem bedenken, dass die Performance der Schreiboperationen auf den **log**-Bereich direkten Einfluss auf die Performance der Datenbank hat. Kauft man seine Datenbank von einem zertifizierten Hardware-

anbieter, kann man diesen Punkt außer Acht lassen, denn die Performanceanforderungen wurden bereits vom Anbieter bedacht. Die verkauften Maschinen sind in diesem Fall außerdem von SAP zertifiziert. Hat man jedoch eigene Hardware, auf der SAP HANA betrieben werden soll, oder verwendet man eigenen Storage, muss man die Performance des Dateisystems im Auge behalten.

Die Größe des Speicherbereichs spielt auch für den Bereich **data** eine Rolle, jedoch in eher untergeordneter Form. Tabelle 2.1 zeigt eine exemplarische Auflistung lesender und schreibender Datendurchsätze in den Speicherbereichen **data** und **log**. Diese Werte sollten mindestens erreicht werden, um SAP HANA produktiv zu betreiben. Es handelt sich dabei um Empfehlungen von SAP.

Speicherbereich	Blockgröße	Testdatei	Schreiben MB/s	Überschreiben MB/s	Lesen MB/s	Latenz µs
log	4 KB	5 GB	–	30	–	1.000
	16 KB	16 GB	–	120	–	1.000
	1 MB	16 GB	–	250	250	–
data	4 KB	5 GB	–	–	–	–
	16 KB	16 GB	40	100	–	–
	64 KB	16 GB	100	150	250	–
	1 MB	16 GB	150	200	300	–
	16 MB	16 GB	200	250	400	–
	64 MB	16 GB	200	250	400	–

Tabelle 2.1 Empfehlungen für den Datendurchsatz der Speicherbereiche »log« und »data« (Quelle: SAP-Hinweis 2762990)

Für Test- und Laborsysteme können diese Werte problemlos unterschritten werden. Zusätzliche Informationen darüber können Sie dem SAP-Hinweis 2762990 entnehmen. Mit den Speicherbereichen **data** und **log** kann die Datenbank nach einem Ausfall bis zur letzten ACID-konform ausgeführten Transaktion wiederhergestellt werden. Weitere Informationen zum Backup- und Recovery-Konzept für SAP HANA finden Sie in Abschnitt 5.5, »Backup und Recovery«.

2.2 Tabellenstrukturen im Speicher von SAP HANA

Datenbanken speichern ihre Datensätze in Tabellen. Diese können unterschiedliche *Tabellenstrukturen* und somit auch unterschiedliche Eigenschaften haben. Welche Strukturen eingesetzt werden, hängt stark vom Datenbestand ab und hat signifikanten Einfluss auf die Leistungsfähigkeit des Systems. Die Struktur beeinflusst zusätzlich die Kompressionsrate einer Datenbanktabelle, denn unterschiedliche Strukturen werden im Arbeitsspeicher anders repräsentiert.

Adressierung von Arbeitsspeicher

SAP HANA setzt zum Speichern von Daten auf zwei Tabellenstrukturen: spaltenbasierte und zeilenbasierte Tabellen. Wir schauen uns zunächst die Repräsentation dieser beiden Strukturen im Arbeitsspeicher an. Den Arbeitsspeicher eines Computers können Sie sich wie eine sehr lange Liste von Einträgen vorstellen, auf die eindimensional zugegriffen wird. Tabelle 2.2 zeigt eine solche Listendarstellung. Geht man die Liste von links nach rechts entlang, springt man immer exakt eine Speicherzelle weiter. Die Speicheradresse erhöht sich also immer um eins.

Zelle 1						Zelle n

Tabelle 2.2 Arbeitsspeicher – dargestellt als Liste

Auf den Arbeitsspeicher wird nicht zellen- oder gar bitweise zugegriffen, sondern in sogenannten *Datenblöcken*. In Tabelle 2.2 werden diese Blöcke durch die verschiedenen Einfärbungen der Zellen repräsentiert. Ein Datenblock umfasst hier also drei Speicherzellen. Wie viele Speicherzellen ein Datenblock umfassen kann, ist abhängig von der eingesetzten Hardware und dem darauf installierten Betriebssystem sowie von der verwendeten Software.

Endet ein Datensatz nicht bündig mit einem Datenblock, muss der nächste Block zusätzlich geladen werden, um den Datensatz vollständig auszulesen. Dies kommt in der Praxis häufig vor und ist auch nicht kritisch. Dieser Umstand muss jedoch bedacht werden, wenn es darum geht, die optimale Performance eines Datenbanksystems zu erreichen.

Beispieldatensatz

Um die Auswirkungen von Speicherblöcken im Datenbankbetrieb zu veranschaulichen, sehen wir uns die Datenverteilung im Arbeitsspeicher anhand eines Beispieldatensatzes an. Tabelle 2.3 zeigt den zur Erklärung verwendeten Testdatensatz mit vier Artikeln, die jeweils eine Artikelnummer und einen Preis haben. Für jeden Artikel werden außerdem der erzielte Umsatz und das Land, in dem der Artikel abgesetzt wurde, aufgeführt.

Artikelnummer	Preis	Umsatz	Land
1	10	500	USA
2	10	600	GER
3	10	700	USA
4	30	800	USA

Tabelle 2.3 Testdatensatz für das Beispiel

2.2.1 Zeilenbasierte Speicherung

Speichert man den Testdatensatz in eine zeilenbasierte Tabelle, werden die einzelnen Datensätze im Arbeitsspeicher aneinandergereiht. Abbildung 2.2 zeigt, wie die Datensätze bei einer zeilenbasierten Tabellenstruktur im Arbeitsspeicher verteilt sind.

Abbildung 2.2 Arbeitsspeicher bei der zeilenbasierten Speicherung

Analytische Abfragen

Möchten Sie jedoch wissen, wie der durchschnittliche Umsatz aller Artikel war, müssen alle vier Datenblöcke vollständig gelesen und verarbeitet werden. Eine solche analytische Abfrage führt in diesem Beispiel zu einem Daten-Overhead von 75 %, denn drei von vier gelesenen Speicherzellen beinhalten für diese Analyse nicht benötigte Daten. Dies zeigt, dass bei analytischen Abfragen zeilenbasierter Tabellen ein großer Daten-Overhead erzeugt werden kann.

Datensatz auslesen

In Abbildung 2.2 sind die Datenblöcke durch die fett gedruckten Rahmen repräsentiert. In diesem Fall ist also jeder Datenblock exakt so groß wie ein Datensatz. Dieses Phänomen tritt in der Praxis selten auf, es dient hier nur der Veranschaulichung. Möchten Sie in dieser vereinfachten Darstellung einen bestimmten Datensatz auslesen, muss genau ein Datenblock gelesen und verarbeitet werden. Das Auslesen einzelner Datensätze geht über eine zeilenbasierte Datenbanktabelle also sehr schnell und produziert geringen bis gar keinen Daten-Overhead. Werden aufgrund der Blockstruktur beim Ausführen einer Transaktion Bestandteile der angeforderten Datensätze geladen, die nicht zur Ausführung benötigt werden, werden diese als *Daten-Overhead* bezeichnet.

Datenkompression

Eine *Datenkompression* in zeilenbasierten Tabellen ist schwierig, denn die aneinandergereihten Daten weisen wenig Dopplungen auf, die im Speicher nebeneinanderliegen. In unserem Beispiel gibt es sogar keine einzige Dopplung. Dies macht den Einsatz von Indizes aufwendig und hält die Kompressionsrate auf einem geringen Niveau. Einen *Index* in einer Datenbank können Sie sich wie ein Inhaltsverzeichnis vorstellen. Er beinhaltet Verweise auf verschiedene Datensätze, um das Suchen und Aufrufen dieser Datensätze zu beschleunigen.

Datensatz hinzufügen

Möchten Sie einen neuen Datensatz in eine zeilenbasierte Tabelle einfügen, wird dieser Datensatz hinter dem letzten Datensatz im Speicher eingefügt und zur späteren Auffindung indiziert. Deshalb geht das Einfügen von Datensätzen in eine zeilenbasierte Tabelle sehr schnell. Es können große Datenmengen in kurzer Zeit in die Tabelle geschrieben werden, ohne viele Systemressourcen zu verbrauchen. Mögliche Einsatzszenarien für zeilenbasierte Tabellen sind daher z. B. die Speicherung von Börsendaten, Positionsdaten großer Fahrzeugbestände oder von Sensordaten, die in Echtzeit geliefert werden.

Tabellenveränderung

Eine Veränderung der zeilenbasierten Tabelle ist jedoch mit großem Aufwand verbunden. Möchten Sie z. B. eine neue Spalte zu Beginn der Tabelle hinzufügen, müssen alle Datensätze im Arbeitsspeicher entsprechend um eine Spalte verschoben werden. Das Gleiche gilt, wenn Sie eine Tabellenspalte wieder entfernen möchten. Deshalb sind Veränderungen von zeilenbasierten Tabellen kostspielig und sollten nach Möglichkeit durch gute Planung im Vorfeld vermieden werden. Sollten trotzdem Anpassungen notwendig sein, sollten Sie diese am besten innerhalb eines Wartungsfensters durchführen, um die Tabellen nicht im Produktivbetrieb zu blockieren.

2.2.2 Spaltenbasierte Speicherung

Spaltenbasierte Tabellen

Einige Charakteristika einer spaltenbasierten Tabelle sind gegensätzlich zu denen einer zeilenbasierten Tabelle. Grundsätzlich können Sie sich spaltenbasierte Tabellen als eine Aneinanderreihung zeilenbasierter Tabellen vorstellen, wobei jede dieser zeilenbasierten Tabellen nur aus genau einer Spalte besteht. Jede Spalte der spaltenbasierten Tabelle wird also durch eine zeilenbasierte Tabelle repräsentiert. Die daraus resultierenden Unterschiede im Speicherverhalten schauen wir uns anhand des aus dem vorangegangenen Abschnitt bekannten Testdatensatzes aus Tabelle 2.3 an. Abbildung 2.3 zeigt, wie die Datenverteilung im Arbeitsspeicher aussieht, wenn Sie eine spaltenbasierte Tabelle für den Testdatensatz einsetzen.

1	2	3	4	10	10	10	30
500	600	700	800	USA	GER	USA	USA

Abbildung 2.3 Arbeitsspeicher bei einer spaltenbasierten Tabelle

Datensatz auslesen

Die einzelnen Datenblöcke werden auch hier wieder durch die einzelnen fett gedruckten Rahmenlinien repräsentiert, es gibt also vier Datenblöcke. Möchten Sie aus dieser Speicherstruktur einen Datensatz auslesen, etwa den für den Artikel mit der Nummer 1, müssen alle vier Datenblöcke gelesen werden. Dies führt zu einem Overhead von 75 %, da auch die Daten der anderen Artikel gelesen werden müssen. Dieser Overhead wird auch als *Rekonstruktionsoverhead* bezeichnet, denn aus allen gelesenen Datenblöcken muss der gewünschte Datensatz anschließend rekonstruiert werden. Wenn es um das Verarbeiten einzelner Datensätze geht, ist eine spaltenbasierte Tabellenstruktur also langsamer als eine zeilenbasierte Tabellenstruktur.

Analytische Abfragen

Anders sieht es bei analytischen Datenbankabfragen aus. Möchten Sie wissen, wie groß der Durchschnittsumsatz aller Produkte ist, müssen Sie in diesem Beispiel nur einen einzigen Datenblock lesen, um die gewünschten Daten zu erhalten, nämlich den mit den Umsatzwerten. Es entsteht also kein Daten-Overhead und auch kein Rekonstruktionsaufwand. Der spaltenbasierte Ansatz verursacht hier also deutlich weniger Aufwand als der zeilenbasierte.

Datenkompression

Eine spaltenbasierte Tabelle erreicht hohe Kompressionsraten. Abbildung 2.4 stellt die Verteilung der Daten des Testdatensatzes aus Tabelle 2.3 im Arbeitsspeicher nach einer Kompression dar. In diesem Beispiel wird eine Reduktion des benötigten Speicherplatzes um ~25 % erreicht.

1	2	3	4	10	30	500	600
700	800	USA	GER				

Abbildung 2.4 Kompression spaltenbasierter Datensätze

Tatsächlich werden in den Datenfeldern zusätzlich noch Indizes hinterlegt, um eine Zuordnung zu ermöglichen. In der Realität ist die Kompressionsrate damit noch deutlich höher, denn typischerweise gibt es eine große

Anzahl an Wiederholungen in Datensätzen. Im Umfeld von Log-Daten sind mit SAP HANA etwa zweistellige Kompressionsraten möglich. Dabei entsprechen 1 GB Log-Daten in SAP HANA 10 GB unkomprimierter Log-Daten außerhalb der Datenbank. Bei anderen Daten erreicht SAP HANA in der Regel hohe einstellige Kompressionsraten.

Weiterführende Informationen zur Datenkompression

SAP hat dem Thema Kompression in SAP HANA den SAP-Hinweis 2112604 gewidmet. Hier werden die unterschiedlichen Kompressionsstrategien von SAP HANA erklärt. Es wird außerdem erläutert, warum es keine feste Kompressionsrate gibt, sondern diese von vielen Parametern abhängt.

Wie Sie gesehen haben, ist es mit spaltenbasierten Tabellen möglich, höhere Kompressionsraten zu erreichen als mit zeilenbasierten Tabellen.

Datensatz hinzufügen

Das Einfügen von Datensätzen in eine spaltenbasierte Tabelle ist hingegen sehr aufwendig, denn die Werte müssen in jede Spalte einzeln eingefügt werden, um anschließend komprimiert und indiziert zu werden. In diesem Punkt sind zeilenbasierte Tabellen also deutlich im Vorteil.

Tabellenveränderung

Eine Veränderung der spaltenbasierten Tabellen ist dagegen mit geringem Aufwand verbunden. Möchten Sie z. B. eine neue Tabellenspalte hinzufügen, können Sie dies einfach durch das Hinzufügen einer neuen zeilenbasierten Teiltabelle erreichen. Es müssen dazu keine bestehenden Datensätze im Arbeitsspeicher verschoben werden. Dasselbe gilt, wenn Sie eine Tabellenspalte entfernen wollen. Es wird lediglich die Tabellenspalte aus dem Speicher entfernt. Tabellenveränderungen sind bei spaltenbasierten Tabellenstrukturen also wesentlich schneller umzusetzen als bei zeilenbasierten Tabellenstrukturen.

Prozessorkerne

Moderne In-Memory-Datenbanken wie SAP HANA verfügen üblicherweise über große Mengen an Arbeitsspeicher. Dieser wird benötigt, um die Datenmengen zu halten und verarbeiten zu können. Um diese Mengen an Daten anbinden zu können, braucht man viele *Speicherbänke*, an die der Arbeitsspeicher angeschlossen werden kann. Diese wiederum benötigen viele Speicher-Controller, die in den *Prozessoren* enthalten sind. Deshalb haben In-Memory-Datenbanken besonders viele Prozessorkerne zur Verfügung. Eine SAP-HANA-Datenbank mit *Hyper-Threading*, die weniger als 50 Prozessorkerne zur Verfügung hat, ist äußert selten. Hyper-Threading ist eine Technologie, die das Ausführen mehrerer paralleler Aufgaben auf einem einzelnen Prozessorkern beschleunigt. Durch mehrere Registersätze erlaubt diese Technologie schnelle Wechsel zwischen den Prozessen.

Parallelisierbarkeit

Um all diese Prozessorkerne auslasten zu können, spielt das Thema Parallelisierung im Umfeld von SAP HANA eine wichtige Rolle. Die Parallelisierbarkeit im Kontext von Datenbanken wurde von Hasso Plattner in einem Paper beleuchtet (Plattner 2009). Darin erläutert er, dass spaltenbasierte Tabellenstrukturen sehr gut zur Parallelisierung geeignet sind. Sie sind in diesem Punkt den zeilenbasierten Tabellen weit überlegen und somit für den Einsatz in In-Memory-Datenbanken prädestiniert.

Vergleich zeilen- und spaltenbasierter Tabellen

Tabelle 2.4 fasst die bisher gewonnenen Erkenntnisse zu den Vor- und Nachteilen zeilen- und spaltenbasierter Tabellen zusammen und stellt die beiden Tabellenstrukturen vergleichend gegenüber.

Eigenschaft	Zeilenbasiert	Spaltenbasiert
Speicherbedarf	höher	geringer
Transaktionen	schneller	langsamer
Datenanalyse	langsamer	schneller
Parallelisierbarkeit	schlechter	besser

Tabelle 2.4 Spaltenbasierte vs. zeilenbasierte Tabellenstrukturen

2.2.3 Delta Merge

Haupteinsatzgebiet: Analyse

SAP HANA spielt seine Stärken vor allem in analytischen Einsatzszenarien aus und besitzt viele Prozessorkerne. Deshalb wurde die Datenbank für spaltenbasierte Tabellen optimiert. Über 90 % der Standardtabellen in SAP HANA sind spaltenbasiert, auch wenn auch zeilenbasierte Tabellen unterstützt werden. In der modernen SAP-Welt gibt es aber auch Einsatzszenarien, in denen in einem kurzen Zeitraum viele Datensätze in die Datenbank geschrieben werden müssen. Um diesen Umstand zu adressieren, hat SAP jede spaltenbasierte Datenbanktabelle in SAP HANA um einen sogenannten *Delta Store* ergänzt.

Delta Store

Der Delta Store ist eine zeilenbasierte unsortierte Tabelle, die den Haupttabellenbereich ergänzt und für alle transaktionalen Operationen, etwa das Einfügen von Datensätzen, verwendet wird. Der Delta Store ist deutlich kleiner als der Haupttabellenbereich. Diese Kombination erlaubt es, das Beste aus den beiden Tabellenstrukturen zu vereinen: die Schnelligkeit einer spaltenbasierten Tabelle bei analytischen Abfragen und deren hohe Kompressionsraten mit der Performance einer zeilenbasierten Tabelle beim Einfügen von Datensätzen. Dieser Aufbau bringt jedoch nicht nur Vorteile mit sich, denn es gibt nun *zwei* Tabellenbereiche, die synchronisiert werden müssen.

Synchronisation

Wenn Sie eine neue Tabelle in SAP HANA erstellen und diese mit Datensätzen füllen, ist zunächst nur der kleinere Delta Store mit Daten befüllt, nicht jedoch der Haupttabellenbereich. Um die Datensätze von einem Bereich in den anderen zu übertragen, die Bereiche also zu *synchronisieren*, wird ein *Burst-Insertion-Verfahren* namens *Delta Merge* verwendet. Bei einem Burst-Insertion-Verfahren wird die Synchronisation nicht sukzessive durchgeführt, sondern alle zu synchronisierenden Datensätze werden auf einmal synchronisiert. Ein Delta-Merge-Vorgang besteht aus drei Zuständen: dem Ausgangszustand, dem Merge-Vorgang und dem Zielzustand.

Ausgangszustand

Abbildung 2.5 zeigt den Ausgangszustand, bevor ein Delta Merge durchgeführt wurde. Alle zu synchronisierenden Daten befinden sich im Deltaspeicher 1. Alle bereits synchronisierten Daten sind im Hauptbereich 1 persistiert. Ausgehend von diesem Zustand, wird die Datensynchronisation durchgeführt.

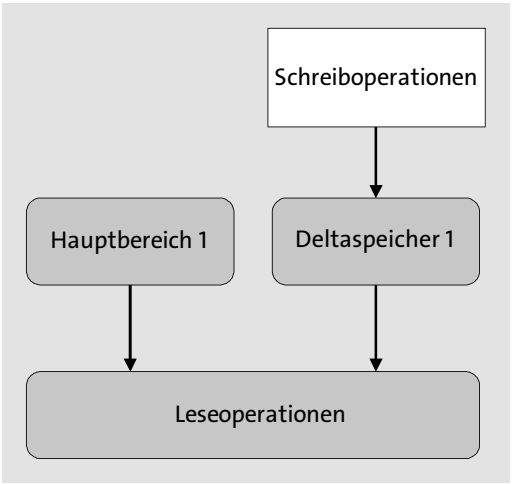

Abbildung 2.5 Ausgangszustand eines Delta Merge

Delta-Merge-Operationen

Abbildung 2.6 stellt alle Tabellen und Datenströme während eines Delta-Merge-Vorgangs dar. Um den alten Deltaspeicher 1 und den Hauptbereich 1 in einen neuen sortierten Hauptbereich 2 zu überführen, werden die sogenannten *Delta-Merge-Operationen* durchgeführt. Diese Operationen umfassen ein Komprimieren sowie ein Sortieren der Datensätze. Hauptbereich 2 wird nach Abschluss des Merge-Vorgangs alle Daten aus Hauptbereich 1 und Deltaspeicher 1 beinhalten. Damit weiterhin Datensätze in die Tabelle geschrieben werden können, übernimmt der Deltaspeicher 2 die Funktion des Deltaspeichers 1. Deltaspeicher 1 kann während des Delta-Merge-Vorgangs für seine ursprüngliche Funktion, das Einfügen neuer Datensätze,

nicht mehr verwendet werden, denn sonst wäre es nicht absehbar, wie lange die Merge-Operationen dauern.

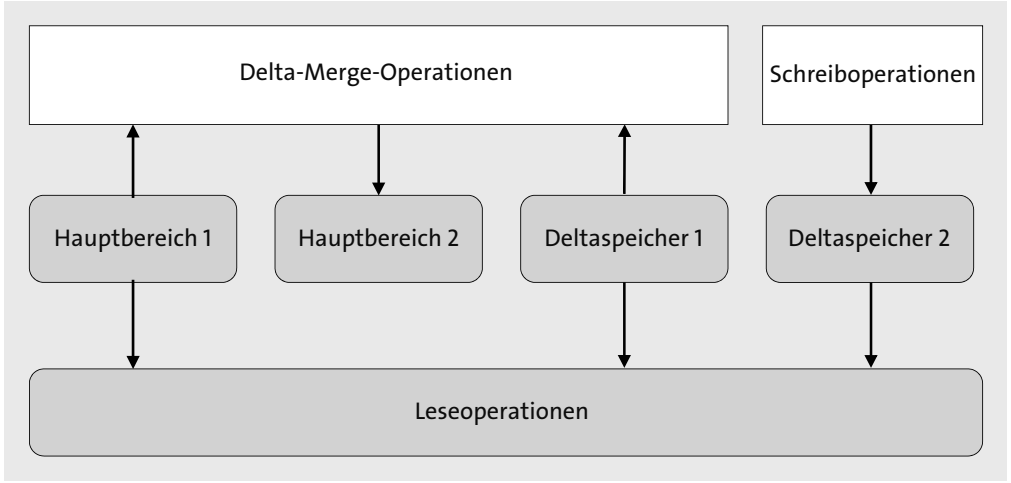

Abbildung 2.6 Zustand während des Merge-Vorgangs

Diese Übernahme der Speicherfunktion durch Deltaspeicher 2 erfolgt, bevor überhaupt Merge-Operationen durchgeführt werden. Um auch während eines Delta-Merge-Vorgangs Datensätze aus der Tabelle abfragen zu können, werden Leseoperationen gleichzeitig auf den Hauptbereich 1, den Deltaspeicher 1 und den Deltaspeicher 2 durchgeführt. Diese drei Speicher beinhalten zusammengenommen alle Datensätze einer Tabelle.

Ist der Delta-Merge-Vorgang abgeschlossen, erhalten Sie den neuen persistenten Zielzustand. Abbildung 2.7 zeigt den Zielzustand, in dem es nur noch den Hauptbereich 2 sowie den Deltaspeicher 2 gibt. Nach dem Erreichen des Zielzustands werden Hauptbereich 1 und Deltaspeicher 1 gelöscht, der Arbeitsspeicher wird also wieder freigegeben. Hauptbereich 2 und Deltaspeicher 2 gelten ab jetzt als neuer Hauptbereich 1 und Deltaspeicher 1. Die Delta-Merge-Operation ist damit abgeschlossen.

Speicherproblematik

Delta-Merge-Operationen können bei großen Tabellen sehr viel Speicher verbrauchen, denn die betroffene Tabelle befindet sich während des Vorgangs zweimal im Arbeitsspeicher. Darüber hinaus sollten Sie bedenken, dass mehrere Delta-Merge-Vorgänge parallel laufen können. Zu diesem Ressourcenbedarf kommt noch der Speicherverbrauch aktueller SQL-Anweisungen aus dem Betrieb der Datenbank hinzu. Dies kann im schlimmsten Fall zu *Unloads* von Tabellen führen. Bei einem Unload werden Tabellen aus dem Arbeitsspeicher entfernt und stehen dann erst mal nur auf der Festplatte zur Verfügung.

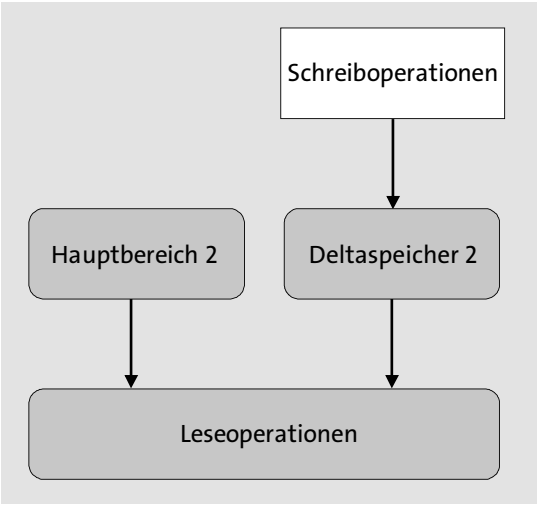

Abbildung 2.7 Zielzustand des Delta Merge

Partitionierung und Delta Merge

Um die Problematik des Speicherverbrauchs beim Delta Merge für große Tabellen zu reduzieren, sollten Sie Tabellen partitionieren. Diese Möglichkeit erläutern wir in Abschnitt 2.5, »Tabellenpartitionierung«. Eine partitionierte Tabelle hat den Vorteil, dass jeder Tabellenabschnitt, also jede *Partition*, ihren eigenen Hauptbereichs- und Deltaspeicher hat.

SAP HANA kann Delta-Merge-Operationen auf den einzelnen Partitionen einer Tabelle durchführen. Sie erfolgen normalerweise jedoch nicht auf allen Partitionen einer Tabelle gleichzeitig, was die Speicherverbrauchsspitzen durch den Delta-Merge-Vorgang deutlich reduziert.

Zeitpunkt für Partitionierungsänderungen [!]

Umpartitionierungen von Tabellen gehen immer mit Delta Merges einher. Deshalb sollten Partitionsänderungen nur in Zeiten durchgeführt werden, in denen die Datenbank keiner großen Last ausgesetzt ist.

Delta-Merge-Statistik

Um einen Überblick über die Delta-Merge-Vorgänge einer SAP-HANA-Datenbank zu bekommen, können Sie eine View aus dem Schema `SYS` aufrufen, in der Statistiken zu den Merge-Vorgängen hinterlegt sind. Sie können diese View mit der folgenden SQL-Anweisung abfragen:

```
SELECT * FROM "SYS"."M_DELTA_MERGE_STATISTICS"
```

Abbildung 2.8 stellt einen Auszug dieser View dar.

MEMORY_MERGE	PASSPORT	START_TIME	EXECUTION_TIME	MOTIVATION	SUCCESS	MERGED_DELTA_RECORDS	LAST_ERROR	ERROR_DESCRIPTION
TRUE		Oct 26, 2020 3:41:36.843 PM	1	AUTO	TRUE	10	0	
TRUE		Oct 26, 2020 3:41:36.843 PM	3	AUTO	TRUE	13,151	0	
FALSE		Oct 26, 2020 3:41:36.843 PM	11	AUTO	TRUE	79	0	
FALSE		Oct 26, 2020 3:41:36.846 PM	12	AUTO	TRUE	2	0	
FALSE		Oct 26, 2020 4:12:24.348 PM	33	AUTO	TRUE	973	0	

Abbildung 2.8 Delta-Merge-Statistik

In Abschnitt 9.1, »Design von Datenbanktabellen in SAP HANA«, erfahren Sie mehr über Datenbankschemata.

Fehler beim Delta Merge

Wenn ein Delta Merge durchgeführt wird – egal, ob automatisch durch SAP HANA oder durch einen manuellen Trigger –, wird immer ein Eintrag in eine Statistiktabelle geschrieben, die Sie sich über diese View ansehen können. Sollte bei einem Delta-Merge-Vorgang ein Fehler aufgetreten sein, finden Sie in der Spalte `LAST_ERROR` der View den Fehlercode und in der Spalte `ERROR_DESCRIPTION` eine Beschreibung des Fehlers. SAP stellt eine Übersicht der möglichen Fehlercodes im »SAP HANA Trouble Shooting and Analysis Guide« zur Verfügung, der für SAP HANA 2.0 SPS05 unter folgender URL abgerufen werden kann: *http://s-prs.de/v834107*.

Es ist nicht nötig, die Delta-Merge-Statistik in regelmäßigen Abständen auf Fehler hin zu prüfen, denn es gibt in SAP HANA entsprechende Alarme (Alerts), die Sie im Fall von fehlerhaften Delta Merges informieren. Ist ein Fehler aufgetreten, ist es jedoch sinnvoll, in die Statistiktabelle zu schauen, um weitere Informationen zum Fehler zu erhalten. Um nach den Fehlerursachen zu forschen, sollten Sie auch mit Traces arbeiten. Treten Delta-Merge-Fehler auf, müssen Sie dabei im Index Server das Trace-Level der Komponenten `mergedog` und `mergemonitor` auf `info` setzen. Näheres zum Tracing erfahren Sie in Abschnitt 7.3.1, »Traces«.

Spalte »SUCCESS«

In der Spalte `SUCCESS` der Delta-Merge-Statistik-View steht nicht, wie man annehmen könnte, ob ein Delta Merge einen Fehler aufwies oder nicht, sondern es wird lediglich festgehalten, ob überhaupt ein Delta Merge stattgefunden hat. Wenn z. B. der Deltaspeicher leer ist und der `mergedog`-Prozess einen zeitbasierten Delta Merge auf eine Tabelle ausführen lässt, kommt es nicht zu einem Fehler. In der Spalte `SUCCESS` steht in diesem Fall der Eintrag `FALSE`, denn es wurde kein Delta Merge durchgeführt. Die Angabe in der Spalte `SUCCESS` beschreibt also, ob Daten vom Deltaspeicher in die Haupttabelle übertragen wurden. Die Anzahl der übertragenen Datensätze können Sie aus der Spalte `MERGED_DELTA_RECORDS` auslesen.

Lang laufende Delta-Merges

In der Praxis laufen Delta-Merge-Vorgänge meist problemlos ab. Es können jedoch ungünstige Konstellationen auftreten, die Sie kennen sollten. Kritisch wird es z. B., wenn ein Delta-Merge-Vorgang länger dauert als die Zeit, die zwischen zwei Speicherpunkten (*Save Points*) abläuft. SAP HANA setzt

in der Standardeinstellung alle 300 Sekunden einen Speicherpunkt. Sollte ein Delta Merge länger als diese 300 Sekunden dauern, wird das Setzen des Speicherpunkts verzögert. Im schlimmsten Fall kann dadurch das gesamte System zum Stehen kommen. Tritt dieses Problem auf, gibt SAP HANA den Alert 54 aus.

Tabelle 2.5 zeigt die Standardschwellenwerte, um den Alert 54 in der Kritikalität niedrig, mittel oder hoch auszulösen.

Dringlichkeit	Niedrig	Mittel	Hoch
Grenzwert	5 Minuten	10 Minuten	15 Minuten

Tabelle 2.5 Schwellenwerte für die Auslösung des Alerts 54

Informationen zur Ursachenanalyse

Sollte Ihnen dieser Alert angezeigt werden, sollten Sie eine Ursachenanalyse (*Root Cause Analysis*) einleiten. Werfen Sie dazu auch einen Blick in SAP-Hinweis 1977220. Hilfreiche Informationen liefert außerdem der Blogbeitrag »HANA – Savepoint details« von Jens Gleichmann in der SAP Community unter *http://s-prs.de/v834108*. Dort finden Sie detaillierte Erklärungen und Hinweise zur Analyse.

2.3 Kommmunikationsschnittstellen

Eine Datenbank ist in einer Systemlandschaft mit vielen unterschiedlichen Akteuren verbunden. Im Fall von SAP HANA sind dies häufig Applikationsserver wie etwa der SAP NetWeaver Application Server for ABAP (AS ABAP) oder SAP NetWeaver Application Server for Java (AS Java) und Data Warehouses wie SAP Business Warehouse (SAP BW) oder SAP BW/4HANA. Aber auch andere Datenbankinstanzen können mit einer SAP-HANA-Instanz in Verbindung stehen, z. B. in einem Systemreplikationsszenario (*SAP HANA System Replication*). Hinzu kommen Administrationstools wie SAP HANA Studio, SAP HANA Cockpit oder HDBSQL, die außerhalb der Datenbank laufen. Um mit allen diesen und weiteren Akteuren kommunizieren zu können, unterstützt SAP HANA verschiedene Kommunikationsschnittstellen.

Architektur einer SAP-HANA-Installation

In Abbildung 2.9 sehen Sie die Architektur einer Single-Host-SAP-HANA-Instanz mit der Systemdatenbank und zwei Tenants sowie der SAP HANA XS Engine. Mehr über diese Komponenten eines SAP-HANA-Systems erfahren Sie in Abschnitt 3.1, »Komponenten der SAP-HANA-Plattform«, und

Abschnitt 3.2, »Das Multi-Tenancy-Konzept«. Damit die Komponenten Daten untereinander austauschen können, werden die Protokolle *SQL*, *OLE DB für OLAP-Datenquellen* (ODBO), *Multidimensional Expressions* (MDX) und *xDBC* verwendet. xDBC steht dabei für den Zugriff auf den Datenbank-Client mit den Technologien *Java Database Connectivity* (JDBC) und *Open Database Connectivity* (ODBC).

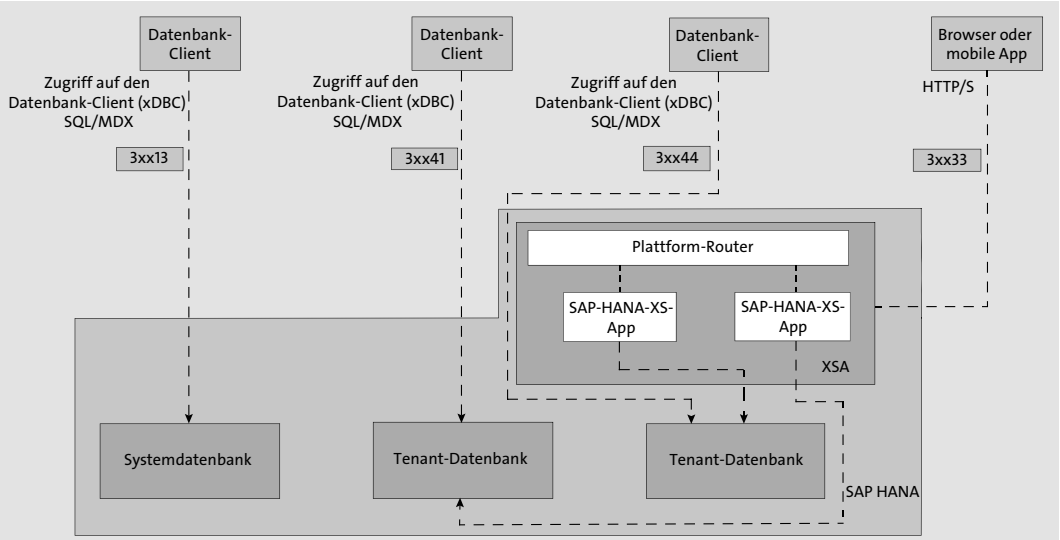

Abbildung 2.9 Architektur einer SAP-HANA-Instanz mit einem Host und zwei Tenants (Quelle: SAP)

Die Kommunikation findet über den SAP-HANA-Datenbank-Client statt, der dafür auf dem Datenbank-Host installiert sein sollte. Es ist auch möglich, eine lokale Instanz des Datenbank-Clients zu verwenden, etwa auf Ihrem Computer. In jedem Fall können Sie davon ausgehen, dass Ihre SAP-HANA-Datenbank immer auch einen Client beinhaltet. In den folgenden Abschnitten gehen wir auf die verschiedenen Schnittstellentechnologien und deren Verwendung im Kontext von SAP HANA ein. Weitere Informationen zum SAP-HANA-Datenbank-Client finden Sie in Abschnitt 3.1.6, »SAP HANA Client«.

2.3.1 ODBC

ODBC ist eine standardisierte Datenbankschnittstelle. Sie wird nicht nur von SAP HANA verwendet, um mit den angebundenen SAP-Systemen zu kommunizieren, sondern z. B. auch von Microsoft SQL Server oder Oracle-Datenbanken. Weil ODBC ein offener Standard ist, gibt es bereits viele Bi-

bliotheken zur Entwicklung von Schnittstellen in Programmiersprachen wie C++, Java etc. Diese Bibliotheken ermöglichen es Entwickler*innen, auf die Datenbank zuzugreifen, ohne dabei auf Datenbankspezifika eingehen zu müssen. Die ODBC-Schnittstelle fungiert dabei als Übersetzerin zwischen der Programmiersprache und der Datenbank. Es werden z. B. Rückgabewerte von SQL-Anweisungen in ein für die jeweilige Programmiersprache benötigtes Format übersetzt und so nutzbar gemacht.

ODBC und SDS

ODBC-Verbindungen zur SAP-HANA-Datenbank können mit dem Verschlüsselungsprotokoll *Secure Sockets Layer* (SSL) bzw. dessen Nachfolgetechnologie *Transport Layer Security* (TLS) verschlüsselt werden. Dies ist u. a. dann interessant für Sie, wenn Sie *Streaming Analytics* für SAP HANA (vormals *SAP HANA Smart Data Streaming*) verwenden möchten, eine Option, die ODBC-Verbindungen unterstützt. Mehr über Streaming Analytics erfahren Sie in Abschnitt 10.2.4, »SAP HANA Streaming Analytics«.

Generell sollten solche Umstellungen immer in Absprache mit SAP erfolgen, um sicherzugehen, dass SAP Support für das eingesetzte Produkts leistet. So können auch viele potenzielle Fehler schon im Vorfeld vermieden werden.

SAP Analysis for Microsoft Office

Ein weiteres Einsatzbeispiel für die ODBC-Schnittstelle ist Microsoft Excel. Mit diesem Tabellenkalkulationsprogramm können Sie direkt auf Datenbanktabellen zugreifen, wenn Sie das Werkzeug *SAP Analysis for Microsoft Office* verwenden. Dieses mit SAP BW ausgelieferte Reporting-Tool ist in Microsoft Excel integriert und bietet Funktionen zur Analyse von Daten und zum Design von Berichten. Damit es auf die SAP-HANA-Datenbank zugreifen kann, müssen Sie den SAP-HANA-ODBC-Treiber für Windows installieren.

Administration und ODBC

Aber nicht jedes SAP-Tool unterstützt automatisch auch die ODBC-Schnittstelle von SAP HANA. Die Administrationswerkzeuge *SAP HANA Studio* und *SAP HANA Cockpit* setzen z. B. auf die JDBC-Schnittstelle, auf die wir im folgenden Abschnitt näher eingehen. SAP HANA Studio ist das ursprünglich von SAP bereitgestellte Administrationswerkzeug für SAP HANA. Es soll strategisch durch das SAP HANA Cockpit ersetzt werden, das aktuell jedoch noch nicht den vollen Funktionsumfang des SAP HANA Studios anbietet. Für das SAP HANA Cockpit wurden dafür einige neue Funktionen implementiert, die es im SAP HANA Studio nicht gab und die dort auch nicht mehr eingeführt werden. Deshalb findet man heute noch viele SAP-HANA-Umgebungen, in denen beide Werkzeuge zum Einsatz kommen. Eine Ein-

führung in die beiden Werkzeuge erhalten Sie in Abschnitt 6.1, »SAP HANA Studio«, und Abschnitt 6.2, »SAP HANA Cockpit«.

Fehleranalyse

Um Fehler bei der Verwendung der ODBC-Schnittstelle analysieren zu können, bietet SAP das Kommandozeilenwerkzeug `hdbodbc_cons` an. Damit können Sie die Kommunikation über die Schnittstelle in einer Log-Datei protokollieren. Abhängig davon, welches Trace-Level Sie für die Protokollierung verwenden, unterscheidet sich die Ausführlichkeit des Ergebnisses. Je höher das Level ist, desto mehr Informationen werden protokolliert. Beachten Sie dabei jedoch auch, dass bei einem hohen Trace-Level mitunter sensible Informationen in den Log-Dateien landen können. Deshalb ist es wichtig, alle Log-Dateien nach erfolgreicher Fehlersuche unwiederbringlich zu löschen. Sie sollten ein Tracing grundsätzlich nur während einer Fehlersuche aktivieren und niemals einfach so mitlaufen lassen.

Wichtige Kommandos

Im Verzeichnis **/hana/shared/<SID>/hdbclient** finden Sie das Tool `hdbodbc_cons`. Sie sollten immer mit dem Administrationsbenutzer <sid>adm angemeldet sein, wenn Sie es ausführen wollen. In Tabelle 2.6 sehen Sie die wichtigsten Befehle, die Sie in dem Kommandozeilenwerkzeug verwenden können.

Kommando	Beschreibung
`hdbodbc_cons show config`	Zeigt die Konfiguration.
`hdbodbc_cons trace sql on`	Startet den SQL Trace.
`hdbodbc_cons trace sql off`	Stoppt den SQL Trace.
`hdbodbc_cons trace api on`	Startet den ODBC Trace.
`hdbodbc_cons trace api off`	Stoppt den ODBC Trace.
`hdbodbc_cons show all`	Zeigt alle aktiven Traces.
`hdbodbc_cons -h`	Ruft die Hilfeseite auf.

Tabelle 2.6 Die wichtigsten Kommandos für »hdbodbc_cons«

Eine vollständige Dokumentation der Befehle erhalten Sie, wenn Sie die Hilfeseite von `hdbodbc_cons` abrufen.

Konfigurationsbeispiel

Wenn Sie den ODBC Trace mit dem Befehl `hdbodbc_cons trace api on` starten und sich anschließend die Konfiguration des ODBC-Kommandozeilenprogramms mit dem Befehl `hdbodbc_cons` show all anzeigen lassen, sollten Sie die in Abbildung 2.10 gezeigte Ausgabe erhalten. Hier sehen Sie unter **Trace file name** den Dateinamen des Traces. Dieser wird durch die Variable `%p`

mit der Prozess-ID des jeweiligen Schnittstellenaufrufs versehen und kann beliebig umbenannt werden.

```
Configuration:
  Trace file name    : sqldbctrace-%p.prt
  Trace flags        : a
    ODBC trace         : disabled
    Short trace        : disabled
    Debug trace        : disabled
    CSE trace          : disabled
    SQL trace          : enabled, DEBUG level
    Distribution trace : disabled
    Packet trace       : disabled
    Flush immediately to disk : disabled

Settings:
  Update count : 9
  Total size   : 26656
    equivalent to 100 process-specific parts.
  Version flag : 1
  Forced re-read of global configuration with last update.

  Process  Update count  Flags
  ---------------------------------------------------------
```

Abbildung 2.10 Anzeige der ODBC-Konfiguration in »hdbodbc_cons«

Die Trace-Dateien finden Sie im Verzeichnis **/hana/shared/<SID>/HDB<Instanznummer>**, in dem sie standardmäßig gespeichert werden. Sie lassen sich z. B. mit dem Texteditor `vi` (siehe auch Abschnitt 3.6.1, »Linux-Grundlagen«) oder mit einem anderen Editor öffnen und durchsuchen. Sie sind menschenlesbar.

Weiterführende Informationen zur Fehleranalyse

Weiterführende Informationen zur Fehleranalyse bei der ODBC-Kommunikation finden Sie im SAP Help Portal unter *http://s-prs.de/v834109*.

ODBC setzt auf dem Protokoll *SQL Database Connectivity* (SQLDBC) auf. Daher kann es mitunter nötig werden, auch eine Fehleranalyse in den Teilen der Kommunikation durchzuführen, die auf SQLDBC basieren. Auf dieses Protokoll gehen wir daher in Abschnitt 2.3.3, »SQLDBC«, ausführlicher ein.

2.3.2 JDBC

JDBC ist eine standardisierte Schnittstelle für die Kommunikation zwischen Datenbanken und Java-Anwendungen. Der ursprüngliche Entwickler dieser Schnittstelle war Sun Microsystems und wurde von Oracle übernommen. JDBC ist auf relationale Datenbanken ausgerichtet und wird mit der Java Standard Edition ausgeliefert. Die Schnittstelle wird nicht nur von Oracle-Datenbanken, sondern auch von SAP HANA, MySQL und Microsoft Azure unterstützt.

Java als Basis

Um JDBC verwenden zu können, werden auf der Client-Seite entsprechende Treiber benötigt. Diese gibt es sowohl für Linux als auch für Microsoft Windows. Der große Vorteil von JDBC liegt darin, dass Java-Entwickler*innen sich keine Gedanken über die verwendete Datenbank und ihre Eigenheiten machen müssen. Die Übersetzung der Java-Befehle in die Sprache der jeweiligen Datenbank wird von JDBC übernommen. Die Ergebnisse der Datenbankabfragen übersetzt JDBC vom jeweiligen Format der Datenbank wiederum zurück in die für Java verwendbaren Formate. Somit können Sie ohne tiefere Datenbankkenntnisse trotzdem alle unterstützten Datenbanken für Ihre Java-Applikationen verwenden. JDBC setzt im Gegensatz zu ODBC und fast allen anderen Protokollen, die im Umfeld von SAP HANA verwendet werden, nicht auf SQLDBC auf, sondern ist ein eigenständiges Protokoll.

[zB]

Eclipse

Ein Beispiel für eine Java-Applikation, die über JDBC auf SAP HANA zugreift, ist *Eclipse*. Diese integrierte Entwicklungsumgebung wird im SAP-Umfeld häufig eingesetzt, u. a. basiert das SAP HANA Studio auf Eclipse. Es gibt zahlreiche Plug-ins für Eclipse für verschiedene Anwendungsfälle, u. a. auch solche für Oracle und MySQL, die die JDBC-Schnittstelle für den Datenbankzugriff verwenden.

Sicherheit

Datenbanken enthalten in der Unternehmenswelt sehr häufig kritische Datensätze. Diese können unterschiedlichster Art sein: technische Zeichnungen, Personaldaten etc. Deshalb sollten Sie darauf achten, grundsätzlich nur ausreichend verschlüsselte Datenbankverbindungen zu verwenden. JDBC-Verbindungen zwischen der SAP-HANA-Datenbank und einem Client können mittels TLS verschlüsselt werden. SAP HANA unterstützt für JDBC außerdem eine Zertifikatsvalidierung, die zwingend erforderlich ist, um *Man-in-the-Middle-Angriffe* zu erschweren. Eine Zertifikatsvalidierung muss immer von beiden Kommunikationspartnern unterstützt werden, um eine optimale Sicherheit gewährleisten zu können.

[»]

Man-in-the-Middle-Angriff

Bei einem Man-in-the-Middle-Angriff befindet sich der Angreifer auf dem Kommunikationspfad zwischen den beiden Daten austauschenden Parteien. Beiden Parteien spielt er vor, der jeweils andere Partner zu sein. So werden zwar die Datenstrecken zwischen den Parteien und dem Angreifer verschlüsselt, er selbst kann aber die gesamte Kommunikation abhören und auch manipulieren.

SAP HANA Cockpit

Das wichtigste Administrationswerkzeug im Umfeld von SAP HANA ist das SAP HANA Cockpit. Es verwendet die JDBC-Schnittstelle, um mit der SAP-HANA-Datenbank zu kommunizieren. Sowohl der SAP HANA Cockpit Server als auch die SAP-HANA-Datenbank unterstützen eine Zertifikatsvalidierung. Somit ist dieser zentrale Knotenpunkt vor Man-in-the-Middle-Angriffen weitestgehend geschützt. Sie müssen darauf achten, dass eine Kommunikationsverschlüsselung und auch die dafür so wichtige Zertifikatsvalidierung eingeschaltet sind. Dies ist nicht standardmäßig der Fall.

SAP HANA Studio

Neben dem SAP HANA Cockpit wird aber auch das SAP HANA Studio noch eingesetzt. Daher muss auch bei dessen Einsatz die Kommunikationssicherheit gewährleistet sein. Auch im SAP HANA Studio können Sie dazu eine Zertifikatsvalidierung verwenden.

Fehleranalyse

Treten Fehler im Kontext der JDBC-Verbindungen auf, können Sie eine Fehleranalyse in der Datei **ngdbc.jar** durchführen. Diese ist Teil des SAP-HANA-JDBC-Treibers und sowohl für Windows als auch für Linux verfügbar. Es gibt die Möglichkeit, einen Trace-Vorgang über die Konsole zu starten, was den Vorteil hat, dass die Anwendung, die per JDBC angebunden ist, dazu nicht gestoppt oder neu gestartet werden muss. Da JDBC Teil des SAP HANA Clients ist, können Sie das Tracing bei auftretenden Problemen direkt in der Linux-Konsole starten, wenn Sie mit dem Benutzer `<sid>adm` angemeldet sind (siehe auch Abschnitt 3.6.1, »Linux-Grundlagen«). Die Datei **ngdbc.jar** befindet sich im Verzeichnis Ihres SAP HANA Clients. Standardmäßig ist dies das Verzeichnis **/hana/shared/<SID>/hdbclient**.

Wichtige Kommandos

In Tabelle 2.7 sehen Sie die wichtigsten Kommandos zur Fehleranalyse der JDBC-Verbindungen.

Kommando	Beschreibung
`ngdbc.jar show config`	Zeigt die Konfiguration.
`ngdbc.jar trace packet on | off`	Schaltet den *Packet Trace* ein/aus. Dieser protokolliert Kommunikationspakete des JDBC-Protokolls.
`ngdbc.jar trace api on | off`	Schaltet den API Trace ein/aus.
`ngdbc.jar trace on | off`	Startet/stoppt den JDBC Trace.
`ngdbc.jar show all`	Zeigt alle aktiven Traces.
`ngdbc.jar -h`	Ruft die Hilfeseite auf.

Tabelle 2.7 Die wichtigsten Kommandos zur Auswertung der Datei »ngdbc.jar«

Nachdem Sie das JDBC Tracing mit dem Befehl `java -jar ngdbc.jar trace on` gestartet haben, sollten Sie die Konfiguration überprüfen, um zu sehen, ob der Trace auch läuft. Abbildung 2.11 zeigt die Konfiguration, wenn alle Trace-Typen eingeschaltet sind und der Trace gestartet ist. Sieht Ihre Konfiguration ebenso aus, läuft das Tracing. Die Trace-Datei **jdbctrace.prt** kann mit `vi` oder einem anderen Editor geöffnet werden und ist menschenlesbar.

```
Driver version                              : 2.4.70-c50fa87a84ec99c122f60a67b56ebbe08d36cc39
Settings file name                          : /root/.sdb/jdbctracesettings.ini
Shared memory file name                     : /root/.sdb/jdbctrace.shm

Configuration
Trace                                       : Enabled
Trace file name                             : jdbctrace.prt
TRACE CONNECTIONS                           : On
TRACE API                                   : On
TRACE PACKET                                : On
TRACE DISTRIBUTION                          : On
TRACE STATISTICS                            : On
TRACE CLEANERS                              : On
TRACE DEBUG                                 : On
Show plain-text client-side encrypted values : Disabled
Show timestamps                             : Disabled
Trace file size                             : Unlimited
Stop on error                               : Disabled
Performance trace                           : Disabled
Performance trace file name                 : jdbcperformancetrace.prt
```

Abbildung 2.11 JDBC-Trace-Konfiguration bei eingeschaltetem Tracing

Tiefer in das JDBC Tracing einsteigen

Um tiefer in das Thema JDBC Tracing einzusteigen, sollten Sie die »SAP HANA Client Interface Programming Reference for SAP HANA Platform« lesen, im Speziellen den Abschnitt »Trace a JDBC Connection« (siehe *http://s-prs.de/v834110*).

2.3.3 SQLDBC

SQL Database Connectivity (SQLDBC) ist eine vom SAP HANA Client unterstützte Laufzeitbibliothek, die sehr performant arbeitet. Ihr Einsatzzweck ist die Entwicklung von Datenbank-Interfaces. Daher bildet SQLDBC die Grundlage für fast alle Schnittstellen des SAP HANA Clients.

Grundlage für die weiteren Schnittstellen

In Abbildung 2.12 sehen Sie die unterstützten Schnittstellen und ihr Zusammenspiel im SAP HANA Client. Neben den Schnittstellen, die SQLDBC als Grundlage verwenden, gibt es auch Applikationen, die direkt auf dieser Technologie aufbauen: SAP S/4HANA, SAP ERP und HDBSQL. Sie haben gemeinsam, dass sie von SAP entwickelt, ausgeliefert und gewartet werden.

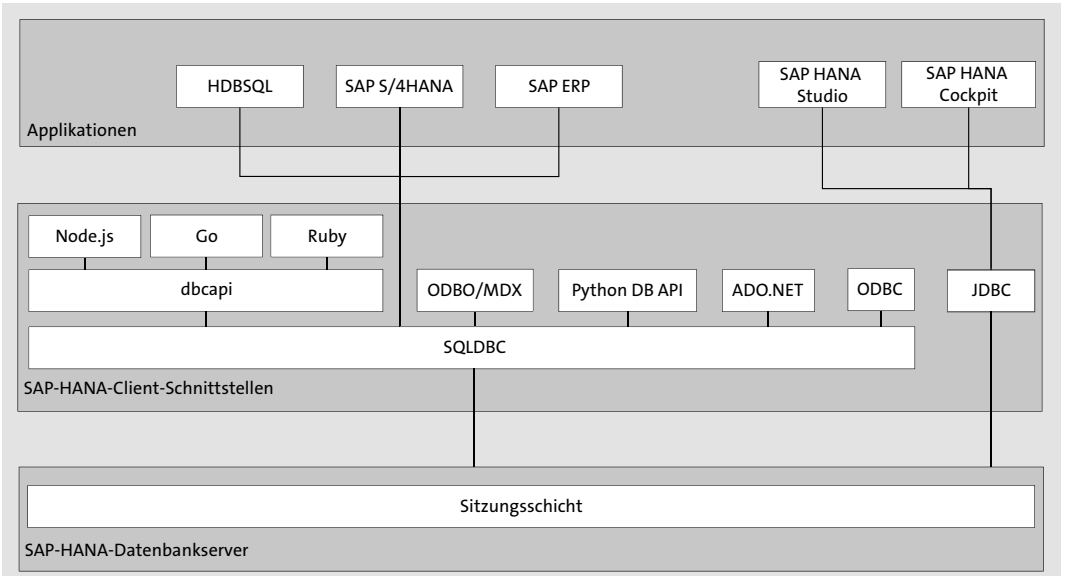

Abbildung 2.12 SQLDBC und die unterstützten Schnittstellen und Applikationen im SAP HANA Client (Quelle: SAP)

HDBSQL

HDBSQL ist ein Werkzeug, das über die Linux-Konsole gestartet werden kann. Es ermöglicht eine direkte Kommunikation mit der Datenbank. Somit kann HDBSQL sehr gut in Linux-Skripten verwendet werden, um unterschiedlichste Datenbankaufgaben zu automatisieren.

Achten Sie darauf, dass Sie beim Aufruf von HDBSQL niemals Ihr Passwort als Parameter mitgeben, denn sonst landet dieser in der Aufrufhistorie (*Bash History*) und ist damit auch für andere Personen zugänglich. Stattdessen rufen Sie HDBSQL ohne den Parameter `-p` auf, was dazu führt, dass Sie zur Passworteingabe aufgefordert werden.

- falsch: `hdbsql -n localhost -u SYSTEM -p <Passwort>`
- richtig: `hdbsql -n localhost -u SYSTEM`

Für die Authentifizierung in Skripten können Sie den SAP-HANA-Benutzerspeicher `hdbuserstore` verwenden. Diesen stellen wir in Abschnitt 6.3, »HDBSQL«, vor.

Fehleranalyse

Da SQLDBC eine Kernkomponente des SAP HANA Clients ist, müssen Verbindungen über diese Schnittstelle fehlerfrei funktionieren. Ansonsten kann es zu schwerwiegenden Kommunikationsproblemen kommen. In diesem Fall kann es passieren, dass Sie den SQLDBC Trace verwenden müs-

sen, um die Ursache für das Fehlverhalten der SQLDBC-Schnittstelle oder einer darauf aufbauenden Schnittstelle zu finden.

Behalten Sie dabei stets im Hinterkopf, dass das Tracing von SQLDBC-Verbindungen die Geschwindigkeit der SAP-HANA-Datenbank signifikant verschlechtert. Außerdem sollten Sie beachten, dass alle Informationen des SQLDBC Traces weder anonymisiert noch pseudonymisiert übermittelt werden. Dies bedeutet, dass sensible Informationen in den Trace-Dateien landen können. Löschen Sie deshalb nach beendeter Fehlersuche alle Trace-Dateien vollständig.

»hdbsqldbc_cons«

Zur Fehleranalyse enthält der SAP HANA Client die Konsolenanwendung `hdbsqldbc_cons`. Die zur Verfügung stehenden Trace-Stufen sind `distribution`, `packet`, `debug` und `sql`. Die Auswahl der passenden Trace-Stufe erleichtert Ihnen die Fehlersuche, da Sie damit die Anzahl der Log-Einträge in der Trace-Datei einschränken können.

Wichtige Kommandos

Auch hdbcsqldbc_cons starten Sie mit dem Benutzer `<sid>adm`. Sie finden das Programm im Verzeichnis Ihres SAP HANA Clients, standardmäßig also unter **/hana/shared/<SID>/hdbclient**. Tabelle 2.8 zeigt Ihnen die wichtigsten Befehle für das Tracing.

Kommando	Beschreibung
`hdbsqldbc_cons show config`	Zeigt die Konfiguration.
`hdbsqldbc_cons trace sql on`	Startet den SQL Trace.
`hdbsqldbc_cons trace sql off`	Stoppt den SQL Trace.
`hdbsqldbc_cons show all`	Zeigt alle aktiven Traces an.
`hdbsqldbc_cons -h`	Ruft die Hilfeseite auf.

Tabelle 2.8 Die wichtigsten Kommandos für »hdbsqldbc_cons«

Eine vollständige Dokumentation aller Befehle finden Sie auf der Hilfeseite. Alternativ können Sie den SQLDBC Tracing Guide von SAP konsultieren (siehe *http://s-prs.de/v834111*).

Trace-Konfiguration

Nachdem Sie den SQL Trace gestartet haben, können Sie sich mit dem Befehl `show all` die Konfiguration anzeigen lassen. Die Ausgabe sollte ähnlich aussehen wie in Abbildung 2.13.

Unter `Trace file name` sehen Sie den Namen der Trace-Datei. Dieser kann mit der Prozess-ID versehen werden, wenn Sie die Variable `%p` hinzufügen. Die Trace-Dateien finden Sie im Verzeichnis **/hana/shared/<SID>/HDB<Instanznummer>**, in dem sie standardmäßig gespeichert werden. Sie lassen sich mit `vi` oder einem anderen Editor öffnen und durchsuchen.


```
Configuration:
  Trace file name    : test.prt
  Trace flags        : a
    Short trace        : disabled
    Debug trace        : disabled
    CSE trace          : disabled
    SQL trace          : enabled, DEBUG level
    Distribution trace : disabled
    Packet trace       : disabled
    Flush immediately to disk : disabled

Settings:
  Update count : 5
  Total size   : 26656
    equivalent to 100 process-specific parts.
  Version flag : 1
  Forced re-read of global configuration with last update.

  Process  Update count  Flags
  ---------------------------------------------------------
```

Abbildung 2.13 SQLDBC – Konfiguration

SAP S/4HANA und SAP ERP setzen beide auf die SQLDBC-Schnittstelle, um mit SAP HANA zu kommunizieren. Somit wird SQLDBC auch zukünftig eine Kerntechnologie im SAP-Universum bleiben.

Tiefer in SQLDBC einsteigen

Weiterführende Informationen zu SQLDBC finden Sie in der »SAP HANA Client Interface Programming Reference« im Abschnitt »SQLDBC Library« (siehe *http://s-prs.de/v834112*).

2.3.4 ODBO und MDX

Die Datenbanksprache MDX ist Teil der Spezifikation ODBO, die 1997 durch Microsoft eingeführt wurde. Die SAP-HANA-Datenbank unterstützt die Version MDX-2005 dieser Sprache. MDX ist ein offener Standard, der von SAP erweitert wurde, um schneller und effizienter zu sein und um die Anforderungen von SAP HANA zu unterstützen. Bei MDX handelt es sich um ein sehr komplexes Themengebiet. Deswegen gehen wir hier nur auf die Verwendung im Kontext von SAP HANA ein.

Microsoft Excel

Microsoft Excel unterstützt mit seinen Pivot-Tabellen MDX-2005. Sie können mit Excel direkt auf die SAP-HANA-Datenbank zugreifen. Dazu müssen Sie Microsoft Excel in der 64-Bit-Version installiert haben. Zusätzlich müssen Sie den SAP HANA Client für Windows und den SAP HANA Client für Excel jeweils in der 64-Bit-Version installieren, um den *SAP HANA MDX Provider* im Datenverbindungsassistenten von Excel angezeigt zu bekommen. Eine detaillierte Anleitung zur Einrichtung des SAP-HANA-Zugriffs in

Excel finden Sie in dem Blogbeitrag »How To Integrate HANA Database with Excel« von Sumit Patel in der SAP Community unter *http://s-prs.de/v834113*.

Verschlüsselung

ODBO-Client-Verbindungen können kritische Datensätze transportieren. Um diese Verbindungen vor einem unberechtigten Zugriff zu schützen, können die Verbindungen zwischen dem SAP HANA Client und der Anwendung, die auf dem ODBC-Client läuft, mittels TLS verschlüsselt werden. Zertifikate lassen sich auch bei ODBO-Verbindungen validieren. ODBO ist deshalb hinsichtlich der Verbindungssicherheit mit ODBC gleichzusetzen.

[+]

Tiefer in ODBO einsteigen

Um tiefer in die Materie von ODBO und MDX im Kontext von SAP HANA einzusteigen, empfehlen wir Ihnen den Abschnitt »ODBO Application Programming« in der »SAP HANA Client Interface Programming Reference for SAP HANA Platform« (siehe *http://s-prs.de/v834114*).

Falls Sie ODBO/MDX im BI-Umfeld einsetzen wollen, sollten Sie zusätzlich noch einen Blick in den Abschnitt »BI ODBO Connector« im Leitfaden »Business Intelligence« von SAP werfen (siehe *http://s-prs.de/v834115*).

2.4 Warm und Hot Storage

Auf unterschiedliche Datensätze wird in der Regel nicht in der gleichen Frequenz zugegriffen, um sie zu analysieren, zu aktualisieren oder zu erweitern. Datensätze in SAP HANA sind von dieser Regel nicht ausgenommen. Ein grundlegendes Konzept von SAP HANA ist es zwar, alle Datensätze im Arbeitsspeicher zu halten, dennoch bietet die Datenbank auch Möglichkeiten zur Datenhaltung außerhalb des Arbeitsspeichers an.

Datentemperatur-Management

Um diese Funktion bereitstellen zu können, werden Datensätze und Speichermöglichkeiten anhand der sogenannten *Datentemperatur* in vier Bereiche eingeteilt:

- Hot Storage
- Warm Storage
- Cold Storage
- Frozen Storage

Welche Datensätze in welcher Temperaturzone persistiert werden, hängt von der Einstufung der Wichtigkeit dieser Daten ab. Im *Hot Storage* sind

Datensätze gespeichert, die häufig verwendet werden und deshalb schnell verfügbar sein müssen. Werden Daten langzeitarchiviert, werden diese im *Frozen Storage* auf Bandlaufwerken gespeichert. Wann Datensätze vom Hot Storage in Richtung des Cold Storage wandern, hängt von den Datensätzen selbst, ihrer Verwendung und auch dem Budget ab. Somit gibt es keine klaren Trennlinien zwischen den einzelnen Speicherbereichen.

Abbildung 2.14 dient als Übersicht über die vier Temperaturzonen und einige der dafür jeweils zur Verfügung stehenden Strategien, die wir im Folgenden näher erläutern.

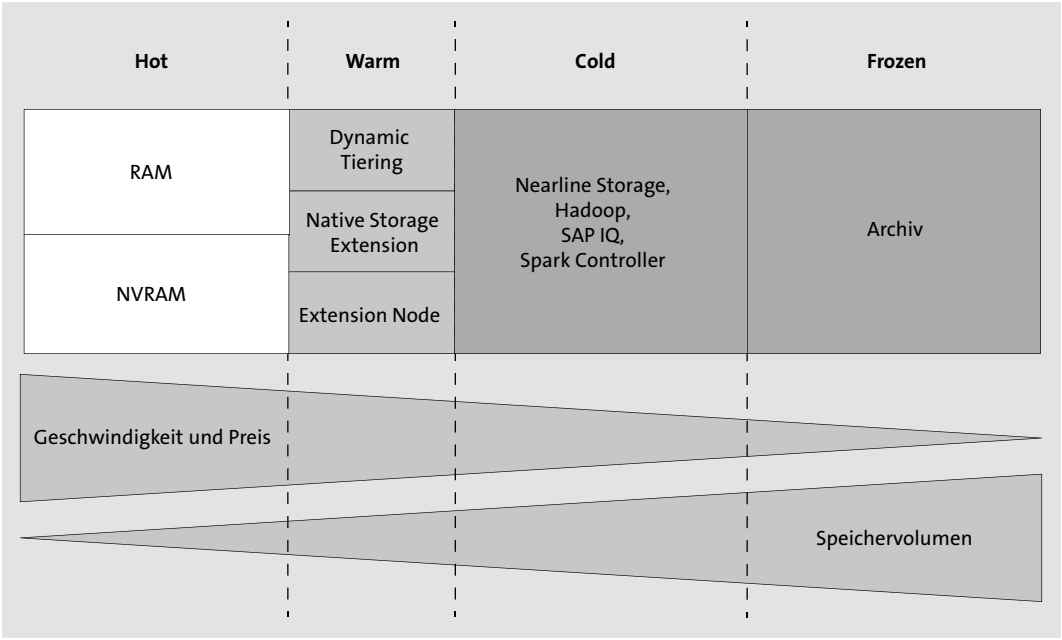

Abbildung 2.14 Hot, Warm, Cold und Frozen Storage in SAP HANA (Quelle: SAP)

Kenngrößen der Speicherkonzepte

Wie Sie anhand der Pfeile in Abbildung 2.14 erkennen können, unterscheiden sich die Temperaturzonen im Wesentlichen durch die drei Kenngrößen Geschwindigkeit, Speichervolumen und Preis. Je schneller der Zugriff auf einen Speicherbereich möglich ist, desto teurer ist er. Außerdem verringert sich in diesem Fall das mögliche Speichervolumen.

[zB]

Kenngrößen eines Frozen Storage

Nehmen wir an, nicht mehr benötigte Datensätze werden auf einem Fujitsu-Bandlaufwerk mit 8.400 TB Speichervolumen archiviert. Die größte SAP-HANA-Datenbank eines zertifizierten Lieferanten kommt lediglich auf

maximal 32 TB Arbeitsspeicher und ist somit um mehr als den Faktor 100 kleiner. Der Geschwindigkeitsunterschied für den Zugriff auf den Frozen Storage ist sogar noch größer. Zur Archivierung eignet sich der Frozen Storage sehr gut, die Zugriffsgeschwindigkeit jedoch ist langsam.

Beispielszenario

In der Praxis werden Sie häufig Szenarien vorfinden, in denen ein Hot und Frozen Storage oder ein Hot, ein Warm und ein Frozen Storage eingesetzt werden, deshalb schauen wir uns ein solches Szenario einmal genauer an. Nehmen wir an, Sie wollen SAP HANA in einem großen Warenhaus zur Speicherung der Produkt- und Verkaufsdatensätze verwenden. Im Regelfall werden diese Daten in regelmäßigen zeitlichen Abständen vom Management des Warenhauses abgefragt, um den Erfolg der Produkte in den einzelnen belieferten Regionen einschätzen zu können. Ladenhüter fliegen aus dem Angebot, und neue Produkte nehmen deren Plätze ein. Der Wert eines Datensatzes verringert sich jedoch im Laufe seiner Lebenszeit. Die Verkaufszahlen von vor fünf Jahren haben nur noch geringen Einfluss auf Managemententscheidungen – ganz im Gegensatz zu den Verkaufszahlen des aktuellen Quartals.

Neben dem Verlust der Relevanz gibt es noch einen zweiten Grund, nicht alle Daten im Speicher zu halten: die schiere Menge an Datensätzen. SAP HANA ist durch den Arbeitsspeicher limitiert, der stets nur zu 50 % mit Datensätzen gefüllt sein darf. Überschreitet die Datenmenge diesen Wert, muss, unabhängig von der Wichtigkeit der Datensätze, eine Lösung gefunden werden. Es gibt grundsätzlich zwei Möglichkeiten, mit zu wenig Platz im Arbeitsspeicher umzugehen. Zum einen könnten Daten ab einem gewissen Alter aggregiert werden, zum anderen kann ihnen eine andere Temperatur zugeschrieben werden.

Aggregation

Unter *Datenaggregation* versteht man das Reduzieren der Datenmenge durch eine Reduktion des Detailgrads. Tagesgenaue Datensätze können z. B. zu wochengenauen Datensätzen aggregiert werden. Tabelle 2.9 enthält einen nicht aggregierten Beispieldatensatz mit tagesgenauer Umsatzauflistung.

Artikel	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
Jeans	50.000 EUR	70.000 EUR	40.000 EUR	30.000 EUR	100.000 EUR
Eis	1.000 EUR	2.000 EUR	3.000 EUR	4.000 EUR	5.000 EUR

Tabelle 2.9 Beispieltabelle mit nicht aggregierten Datensätzen

Anhand dieser Tabelle ließen sich sowohl der Wochenumsatz als auch der Tagesumsatz eines Produkts analysieren. Durch eine Aggregation würde nur der Detailgrad verringert. Tabelle 2.10 zeigt das Ergebnis einer Aggregation auf wochengenaue Datensätze.

Artikel	Kalenderwoche	Umsatz
Jeans	1	290.000 EUR
Eis	1	15.000 EUR

Tabelle 2.10 Aggregation der Datensätze auf Wochengenauigkeit

Auf Basis dieser aggregierten Datensätze können keine tagesgenauen Aussagen mehr getroffen werden. Dafür ist jedoch die benötigte Speichermenge deutlich geringer.

Aggregationsstufen

Je nach Einsatzszenario kann es sinnvoll sein, mehrere Aggregationsstufen zu verwenden, in denen bereits aggregierte Datensätze noch weiter aggregiert werden. Wochengenaue Daten können etwa auf den Monat, das Quartal oder das Jahr aggregiert werden. Die weitere Aggregation reduziert sowohl den Speicherverbrauch als auch den Detailgrad der Daten drastisch.

Temperaturreduktion

Eine andere Möglichkeit zur Problemlösung ist die Reduktion der Temperatur von Datensätzen. Weniger wichtigen Datensätzen wird dabei eine niedrigere Temperatur zugewiesen, um sie auf günstigere Speichermedien auszulagern. Die so kategorisierten Datensätze befinden sich somit nicht mehr direkt im Arbeitsspeicher der SAP-HANA-Datenbank.

Mit dieser Herangehensweise behält man im Gegensatz zur Aggregation den vollständigen Detailgrad der Daten und reduziert trotzdem die Speicherkosten pro Gigabyte Datenbestand. SAP HANA bietet unterschiedliche technische Möglichkeiten zur Temperaturreduktion von hot auf warm an, die wir Ihnen in den folgenden Abschnitten vorstellen:

- SAP HANA Dynamic Tiering
- SAP HANA Native Storage Extension
- SAP HANA Extension Node

Kombination der beiden Ansätze

Technisch lassen sich die Strategien Temperaturreduktion und Aggregation auch miteinander kombinieren. Diese Kombination sollte allerdings genau mit der Verwendung der Datensätze im geschäftlichen Kontext abgestimmt sein.

2.4.1 SAP HANA Dynamic Tiering

SAP HANA Dynamic Tiering ist eine Komponente von SAP-HANA-Datenbanken, mit der sich Datensätze auf einem externen Speicherbereich speichern lassen. Sie unterstützt native SAP-HANA-Anwendungen wie SAP S/4HANA, SAP BW/4HANA und die SAP Business Suite auf SAP HANA. SAP HANA Dynamic Tiering lässt sich in Multi-Tenancy-Szenarien einsetzen, jedoch nicht im *High-Isolation-Modus*. Mehr über das Multi-Tenancy-Konzept und den High-Isolation-Modus erfahren Sie in Abschnitt 3.2.

High-Isolation-Level

Das High-Isolation-Level in SAP HANA ist ein Modus, um die Abgrenzung zwischen mehreren Tenant-Datenbanken zu verbessern. Dabei wird z. B. ein Betriebssystembenutzer für jeden einzelnen Tenant verwendet. Im Gegensatz dazu verwenden die Tenants im Standardmodus alle den Benutzer `<sid>adm` auf Betriebssystemebene. Ziel des High-Isolation-Levels ist es, unterschiedliche Abteilungen eines Unternehmens auf einer SAP-HANA-Datenbank arbeiten lassen zu können, ohne dabei den Grundsatz der Aufgabentrennung (*Segregation Of Duties*) zu verletzen. Alle Administratoren einer Abteilung sollten mit diesem Konzept nur ihre eigenen Tenants administrieren können.

Deployment-Optionen

SAP HANA Dynamic Tiering lässt sich auf zwei Arten installieren:

- Zum einen kann ein *Dedicated Host Deployment* verwendet werden, bei dem eine eigene Host-Maschine für den Dynamic Tiering Server genutzt wird.
- Zum anderen kann ein *Same Host Deployment* verwendet werden, bei dem die SAP-HANA-Datenbank und der Dynamic Tiering Server auf derselben Host-Maschine laufen.

Abbildung 2.15 zeigt ein Dedicated Host Deployment, bei dem der Dynamic Tiering Server einen Standby Server als Backup zur Verfügung hat.

HDBLCM

Da das Dynamic Tiering eine Komponente von SAP HANA ist, kann es nicht ohne eine SAP-HANA-Datenbank eingesetzt werden. Die Komponente muss mit dem *SAP HANA Database Lifecycle Manager* (HDBLCM) installiert werden. Dieses Standardwerkzeug wird bei der Installation von SAP HANA mitinstalliert. Es befindet sich, wenn nicht umkonfiguriert, im Standardverzeichnis **/hana/shared/<sid>/**.

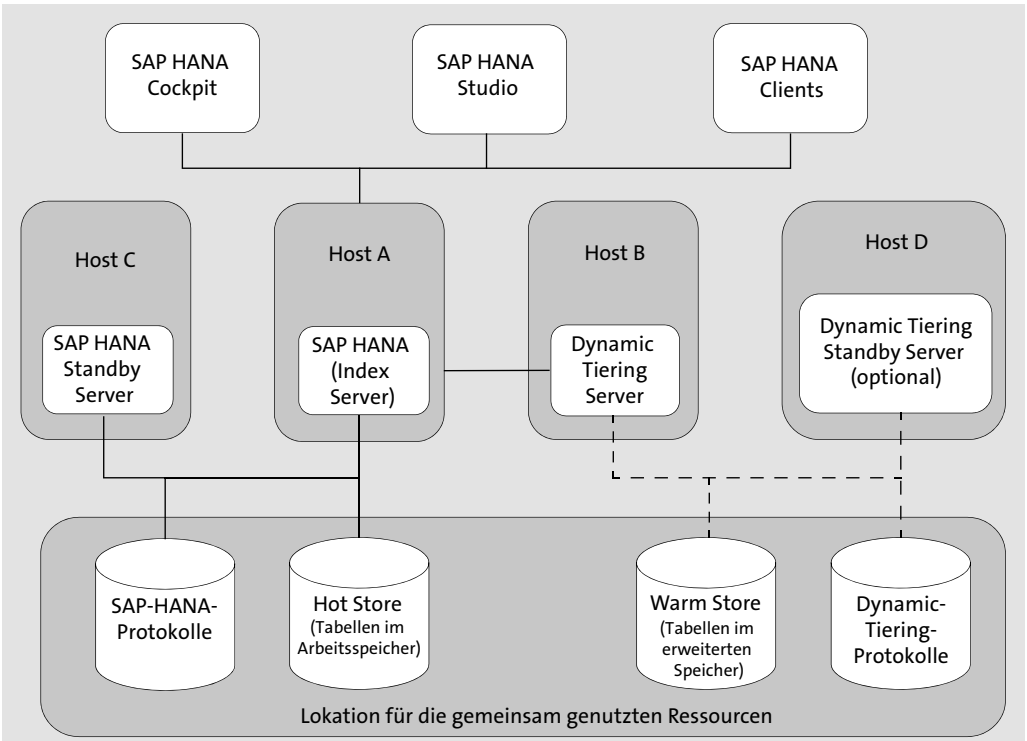

Abbildung 2.15 Beispiellandschaft für den Einsatz von SAP HANA Dynamic Tiering (Quelle: SAP)

Wie eng verwachsen SAP HANA und das Dynamic Tiering sind, können Sie schon bei der Installation erkennen. Das Verzeichnis **/hana/shared** muss auch auf der Dynamic-Tiering-Maschine verfügbar sein, auch wenn es auf einem anderen Host läuft. Dies kann z. B. in einem *Linux Network Filesystem* (NFS) erreicht werden. Beachten Sie in diesem Fall, dass NFS standardmäßig nicht verschlüsselt kommuniziert und damit eine Sicherheitslücke darstellen kann, sollten Sie die Kommunikation nicht zusätzlich absichern.

Lizenzen

Für einige SAP-HANA-Komponenten werden zusätzlich zu der Lizenz für die SAP-HANA-Datenbank weitere Lizenzen benötigt. Das ist auch bei SAP HANA Dynamic Tiering der Fall. Es wird eine eigene Lizenz benötigt, deren Gebühr sich nach der Größe des komprimierten Speicherbereichs berechnet. Diese Lizenz muss in die Systemdatenbank eingespielt werden und nicht in eine Tenant-Datenbank. Alle Tenant-Datenbanken teilen sich dieselbe Lizenz für SAP HANA Dynamic Tiering. Es ist unbedingt notwendig, immer eine gültige Lizenz zu haben. Sollte eine Lizenz auslaufen, kann SAP HANA Dynamic Tiering nicht mehr verwendet werden. Dies kann zu schwerwiegenden Problemen, vor allem im produktiven Betrieb, führen.

Aktuelle Lizenzen abfragen

Eine Übersicht über die gültigen Lizenzen Ihrer SAP-HANA-Datenbank und der von Ihnen verwendeten Komponenten können Sie sich über die View `M_LICENSES` ausgeben lassen. Verwenden Sie dazu folgende SQL-Anweisung:

```
SELECT * FROM "SYS_DATABASE"."M_LICENCES"
```

Tabellenbereiche einer Multistore-Tabelle, die auf einem Dynamic Tiering Server ohne gültige Lizenz liegen, werden nicht automatisch in normale Tabellen umgewandelt. Zugriffe auf die Datensätze in einem Dynamic Tiering Server ohne gültige Lizenz sind in jedem Fall eingeschränkt.

Multistore-Tabelle

Abbildung 2.16 zeigt eine sogenannte *Multistore-Tabelle* mit vier Partitionen. Die Partitionen Hot_P1 und Hot_P2 liegen im Arbeitsspeicher der SAP-HANA-Datenbank, während die Partitionen Warm_P3 und Warm_P4 auf einem Dynamic Tiering Server gespeichert sind. Dies ist das Unterscheidungsmerkmal zu normalen Tabellen, denn diese liegen im Regelfall nur auf einer Maschine.

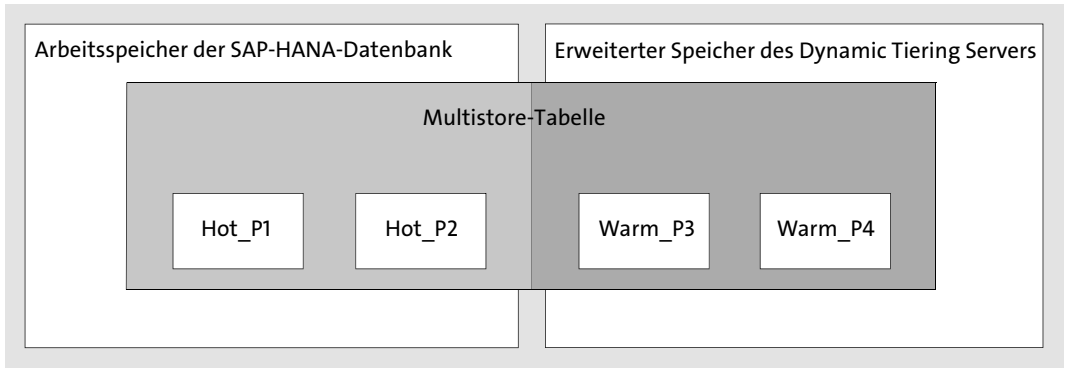

Abbildung 2.16 Beispiel für eine Multistore-Tabelle in SAP HANA (Quelle: SAP)

Das Multistore-Konzept wird nur für spaltenbasierte Tabellen unterstützt. Multistore-Tabellen sind tief in SAP HANA integriert und werden vollständig durch SAP HANA und das SAP HANA Dynamic Tiering verwaltet. Deshalb sehen sie von außen betrachtet wie normale spaltenbasierte Tabellen aus. Alle Datenmanipulationen können mit denselben SQL-Anweisungen wie bei normalen spaltenbasierten Tabellen ausgeführt werden, also z. B. `INSERT`, `UPDATE`, `ALTER` etc. Dies macht die Verwendung von Multistore-Tabellen sehr einfach.

Multistore-Tabelle anlegen

Um eine Multistore-Tabelle mit Partitionen anzulegen, verwenden Sie die SQL-Anweisung aus Listing 2.1.

```
CREATE COLUMN TABLE MY_TABLE (
 a INT, b INT, c INT, PRIMARY KEY (a, b)
                              )
PARTITION BY RANGE (a)(
USING DEFAULT STORAGE (PARTITION 1 <= VALUES < 5, PARTITION 5 <=
 VALUES < 20, PARTITION VALUE = 44, PARTITION OTHERS
                         )
USING EXTENDED STORAGE (PARTITION VALUE = 45));
```

Listing 2.1 Multistore-Tabelle anlegen

Die Partitionen, die über den Zusatz `USING DEFAULT STORAGE` definiert werden, befinden sich im Arbeitsspeicher der SAP-HANA-Datenbank, denn dieser ist der *Default Storage*. Alle Partitionen, die mit dem Zusatz `USING Extended Storage` definiert werden, befinden sich auf dem Dynamic Tiering Server. Die Partitionierung kann außerdem auf Basis unterschiedlicher Datentypen erfolgen. In unserem Beispiel haben wir über den Wert in Spalte a partitioniert, der vom Datentyp Integer ist.

Partition Pruning

SAP HANA unterstützt das sogenannte *Partition Pruning* auch für Multistore-Tabellen. Unter Partition Pruning versteht man im Datenbankenumfeld, dass der FROM- und WHERE-Teil einer SQL-Anweisung ausgewertet werden. Dabei wird darauf geachtet, wo genau sich die zu verarbeitenden Datensätze befinden. So können nicht benötigte Datenpartitionen während der Anweisungsausführung ausgelassen werden. Mit dieser Strategie wird erreicht, dass SQL-Anweisungen auf eine Multistore-Tabelle, die nur Datensätze abfragen, die sich im Arbeitsspeicher der SAP-HANA-Datenbank befinden, nicht langsamer verarbeitet werden als Zugriffe auf eine reguläre spaltenbasierte Tabelle.

Der Dynamic Tiering Server berechnet eigenständig die Zwischenergebnisse, bevor diese an die SAP-HANA-Datenbank übertragen werden. Dies reduziert den Geschwindigkeitsverlust, den eine Übertragung aller zur Ausführung benötigten Datensätze über das Netzwerk bedeuten würde. Vollständig verhindern lässt sich ein Performanceverlust jedoch nicht.

Es ist möglich, Datensätze zwischen dem Extended Storage und dem Default Storage zu verschieben. Dabei müssen Sie jedoch beachten, dass nur vollständige Partitionen verschoben werden können. Sollte sich also der zu verschiebende Datensatz innerhalb einer Partition befinden, muss diese erst umpartitioniert werden.

Unterstützte SAP-HANA-Standards

SAP HANA Dynamic Tiering unterstützt die Standard-Backup-Strategien *Full-Backup*, *Delta-Backup* und *Log-Backup*. Diese Strategien stellen wir Ihnen in Abschnitt 5.5, »Backup und Recovery«, vor. Die *Backint-Schnittstelle* von SAP HANA funktioniert auch mit Datensätzen, die sich auf einem Dynamic Tiering Server befinden. Über diese Schnittstelle kann Drittanbietersoftware zur Erstellung von Backups an SAP HANA angebunden werden.

Zusätzlich wird die SAP HANA System Replication unterstützt. Den Einsatz von SAP HANA Dynamic Tiering sollten Sie daher bereits bei der Planung einer SAP-HANA-Datenbanklandschaft berücksichtigen, um Kosten bei der Verarbeitung großer Datenmengen zu sparen.

2.4.2 SAP HANA Native Storage Extension

Mit SAP HANA 2.0 SPS04 führte SAP das Feature SAP HANA Native Storage Extension (NSE) ein. Damit lassen sich Tabellenspalten als *Seiten* handhaben. Eine Seite ist dabei eine Teilmenge einer Tabellenspalte. Sie erlaubt eine granulare Datenhaltung. Außerdem erlaubt es SAP HANA Native Storage Extension, Festplattenspeicher zur Kapazitätserweiterung zu verwenden, womit Daten aus dem Hauptspeicher auf die Festplatte ausgelagert werden können. Mit dem zum Zeitpunkt der Drucklegung dieses Buches aktuellen Release SAP HANA 2.0 SPS05 wird dieses Feature nur für spaltenbasierte Datenbanktabellen verwendet. Es gibt keine Möglichkeit, es auch für zeilenbasierte Tabellen zu nutzen.

Ladeverhalten spaltenbasierter Tabellen

Standardmäßig lädt eine SAP-HANA-Datenbank alle spaltenbasierten Tabellen und deren Tabellenspalten in den Arbeitsspeicher, nachdem sie gestartet wird. Tabellen-Unloads betreffen auch Tabellen oder Spalten. Einzelne Datensätze oder Gruppen von Datensätzen können somit nicht aus dem Arbeitsspeicher entfernt oder neu geladen werden.

Jede Tabellenspalte hat ein sogenanntes *Preload Flag*, das bestimmt, ob eine Spalte beim Starten der Datenbank in den Arbeitsspeicher geladen wird. Ist dieses Flag für alle Spalten einer Tabelle gesetzt (`preload=true`), wird die gesamte Tabelle geladen. Ist es nicht gesetzt (`preload=false`), wird die entsprechende Tabelle nicht in den Arbeitsspeicher geladen und verbleibt auf der Festplatte. Wird versucht, auf Datensätze einer nicht geladenen Tabelle zuzugreifen, werden nur die benötigten Tabellenspalten vom Datenbank-Storage in den Arbeitsspeicher geladen. Sie können das Ladeverhalten bestimmter Tabellenspalten mit der SQL-Anweisung aus Listing 2.2 abfragen:

```
SELECT SCHEMA_NAME, TABLE_NAME, COLUMN_NAME, PRELOAD
FROM TABLE_COLUMNS
WHERE SCHEMA_NAME = '<Name des Datenbankschemas'
AND TABLE_NAME = '<Tabellenname>'
```

Listing 2.2 Ladeverhalten von Tabellenspalten abfragen

Ressourcen

In Abbildung 2.17 sehen Sie, auf welche Ressourcen SAP HANA NSE zugreift. Es gibt einen vorreservierten Bereich im Arbeitsspeicher der Datenbank, der von SAP HANA NSE dazu genutzt wird, Seiten aus der Persistenzschicht der Datenbank zu puffern. Diesen bezeichnet man auch als *NSE-Cache*. Zusätzlich gibt es einen Speicherbereich auf dem Storage der Datenbank, in dem die Seiten einer Tabelle gespeichert sind. Dieser wird von der Persistenzschicht verwaltet (siehe Abbildung 2.17). Wird auf Datensätze einer Seite zugegriffen, die sich nicht im NSE-Cache befindet, wird diese Seite zunächst aus dem NSE-Datenbereich in den Cache geladen.

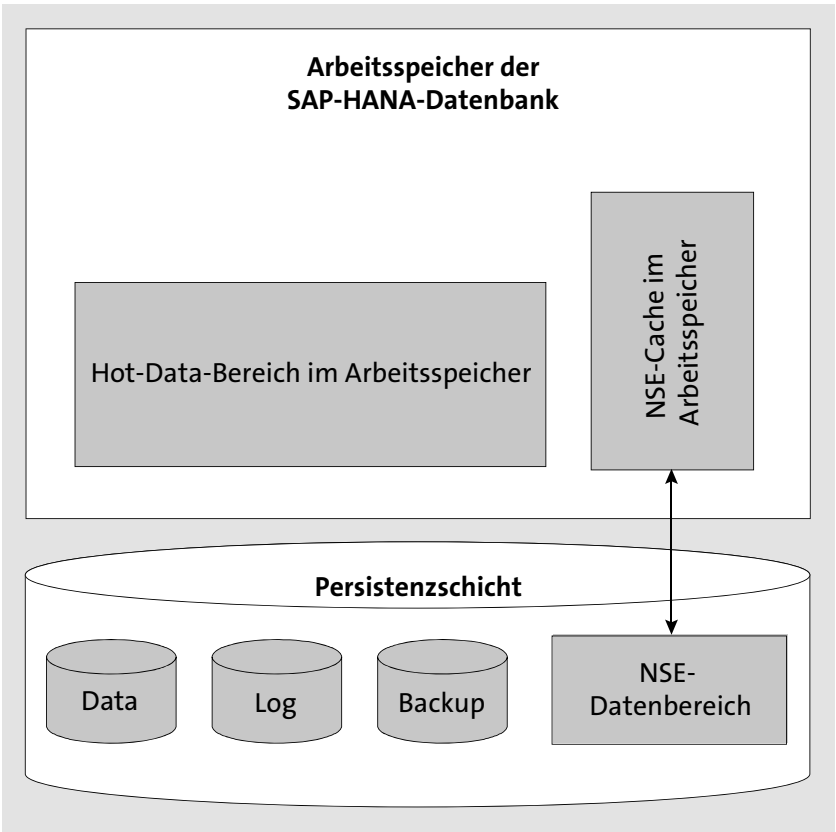

Abbildung 2.17 Verwendete Datenbankressourcen beim Einsatz von SAP HANA NSE (Quelle: SAP)

Ist der Ladevorgang abgeschlossen, werden die Daten zur Verarbeitung verwendet. Dieses Vorgehen ist selbstverständlich langsamer als der Zugriff auf Daten, die bereits im Arbeitsspeicher liegen. Deshalb ist es beim Einsatz von SAP HANA NSE sehr wichtig, einen schnellen Speicherbereich als NSE-Datenbereich zu verwenden.

LRU-Strategie

Datensätze werden erst dann aus dem NSE-Cache gelöscht, wenn der noch frei verfügbare NSE-Cache-Bereich nicht mehr ausreichend groß für eine neue Datenanfrage ist. SAP setzt für den NSE-Cache auf die Cache-Strategie *Last Recently Used* (LRU). Bei dieser Strategie werden immer diejenigen Seiten aus dem Cache gelöscht, auf die die längste Zeit über nicht mehr zugegriffen wurde. Dieses Verhalten sollten Sie bei der Datenpartitionierung im Hinterkopf behalten, denn im schlimmsten Fall werden bei einer homogenen Verteilung von Datenzugriffen sonst ständig Seiten aus dem NSE-Datenbereich in den NSE-Cache geladen. Dies würde den Geschwindigkeitsvorteil, den der Cache bringt, wieder zunichtemachen.

NSE-Cache konfigurieren

Standardmäßig werden 10 % des gesamten Arbeitsspeichers des Datenbankservers als NSE-Cache reserviert. Sie können dies jedoch über die Parameter `max_size_rel` und `max_size` anpassen, die Sie mit folgender SQL-Anweisung beeinflussen können:

```
ALTER SYSTEM ALTER CONFIGURATION ('indexserver.ini', 'system')
 SET ('buffer_cache_cs','max_size_rel') = '10'
WITH RECONFIGURE;
```

Tabellenbereiche auslagern

Um Tabellenbereiche in den NSE-Datenbereich auszulagern, müssen Sie die Tabellen entsprechend konfigurieren. Zunächst werden die Tabellen dazu partitioniert. Die einzelnen Partitionen werden dann entweder als *Page Loadable* oder *Column Loadable* gekennzeichnet. Page-Loadable-Partitionen werden dann automatisch im NSE-Datenbereich gehalten.

Wir schauen uns das an einem Beispiel an. Ausgangsbasis ist der Datensatz in Tabelle 2.11.

Jahr	Produktnummer	Verkäufe
2021	123	1.000
2020	123	12.000
2019	123	12.000

Tabelle 2.11 Beispieldatensatz für die Konfiguration des NSE-Datenbereichs

Im Regelfall werden nur die Daten des aktuellen und des vorangegangenen Jahres benötigt. Deshalb ergibt es Sinn, alle Daten, die älter sind, in den NSE-Datenbereich auszulagern. Führen Sie die SQL-Anweisung aus Listing 2.3 aus, um die Daten aus dem Jahr 2019 auszulagern:

```
CREATE COLUMN TABLE Meine_TABELLE (
  a DATE, b INT, c INT, PRIMARY KEY (a)
                                    )
PARTITION BY RANGE (year(a))
                              (
PARTITION '2021' <= values <'9999' COLUMN LOADABLE, PARTITION '2019'
< values < '2021'   PAGE LOADABLE, PARTITION OTHERS PAGE LOADABLE
                               );
```

Listing 2.3 Tabelle partitionieren und einzelne Partitionen als Page Loadable markieren

Tiefer in SAP HANA NSE einsteigen

Um tiefer in das Thema SAP HANA NSE einzusteigen, sollten Sie den SAP-Hinweis 2799997 lesen. Außerdem ist das Kapitel »SAP HANA Native Storage Extension« im »SAP HANA Administration Guide« relevant. Empfehlenswert ist außerdem der Blogartikel »SAP HANA Native Storage Extension (NSE) – Increase HANA Data Capacity With Warm Storage« von Jeetendra Kapase in der SAP Community unter *http://s-prs.de/v834116*.

2.4.3 SAP HANA Extension Node

SAP HANA Extension Node ist eine Strategie zur Speicherung warmer Datensätze, die auf dem *Scale-out-Prinzip* basiert. Bei einem Scale-out wird eine Datenbank erweitert, indem weitere Maschinen bereitgestellt werden. In Gegensatz dazu wird bei einem *Scale-up* die Hardware der Datenbankmaschine erweitert, indem z. B. größere Arbeitsspeicherriegel oder Prozessoren mit mehr Kernen eingeführt werden. Für den Einsatz von SAP HANA Extension Node ist eine Scale-out-Landschaft zwingende Voraussetzung.

Scale-out-Landschaft

Eine solche Landschaft mit einem Erweiterungsknoten (*Extension Node*) wird in Abbildung 2.18 veranschaulicht. Eine Scale-out-Landschaft hat immer einen Leitknoten (*Master Node*) und mindestens einen Folgeknoten (*Slave Node*). Der kleinste Aufbau einer Scale-out-Landschaft bestünde somit aus einem Leitknoten und einem Folgeknoten als Extension Node. Die Anzahl an Folge und Stand-by-Knoten kann jedoch deutlich größer sein. Ein Stand-by-Knoten übernimmt die Aufgabe eines Leit- oder Folge-

knotens, sollte dieser ausfallen. Einzig die Anzahl der Extension Nodes ist laut SAP-Empfehlung auf einen beschränkt. Das Thema Ausfallsicherheit behandeln wir in Abschnitt 3.5, »Hochverfügbarkeit«, ausführlicher.

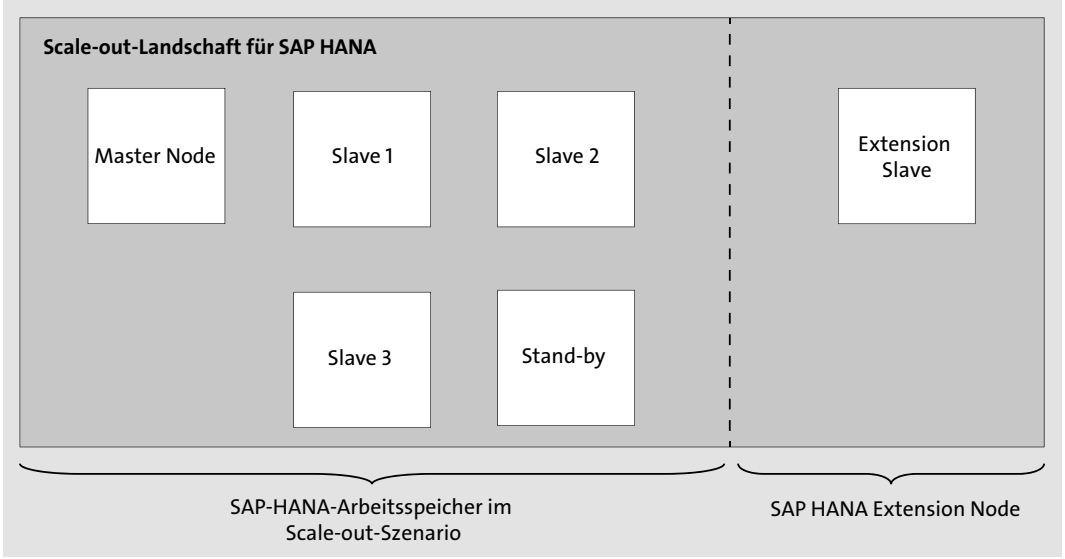

Abbildung 2.18 Scale-out-Landschaft für den Einsatz von SAP HANA Extension Node (Quelle: SAP)

Mehr als ein Extension Node

Technisch gäbe es die Möglichkeit, mehr als einen Extension Node zu verwenden. Einen Hinweis darauf finden Sie in den Frequently Asked Questions (FAQ), die dem SAP-Hinweis 2486706 angehängt sind. Dort wird erklärt, dass pro SAP-HANA-Scale-out-Landschaft jeweils nur ein Extension Node vorgesehen ist, weil dieser im Regelfall zur Datenhaltung ausreicht. Eine größere Anzahl an Extension Nodes ginge auch mit einem höheren Datenmanagement-Overhead einher, was die Latenz beeinflussen würde. Sollten Sie einen weiteren Extension Node benötigen, können Sie zu dieser Anfrage ein Kundenticket bei SAP öffnen.

Kapazität des Extension Nodes

Extension Nodes können mehr Datensätze speichern als eine normale SAP-HANA-Datenbank mit gleich viel Arbeitsspeicher laut SAP-Spezifikationen. SAP empfiehlt, niemals mehr als 50 % des zur Verfügung stehenden Arbeitsspeichers mit Daten zu füllen, damit immer noch genügend Ressourcen für Delta Merges und das Ausführen von SQL-Anweisungen zur Verfügung stehen. Setzen Sie z. B. eine SAP-HANA-Datenbank mit 8 TB Arbeitsspeicher ein, dürfen davon maximal 4 TB mit Daten belegt sein. In

einem Extension Node hingegen ist das Größenverhältnis zwischen Datenmenge und Arbeitsspeicher ein anderes. Hier befinden Sie sich noch innerhalb des von der SAP-Spezifikation festgelegten Rahmens, wenn der *Overload-Faktor* zwischen 2 und 4 liegt.

Overload-Faktor

Ein Overload Faktor von 2 entspricht 2 × 50 % des gesamten Arbeitsspeichers. Ein Overload-Faktor von 4 hingegen entspricht 4 × 50 %, also 200 % des Arbeitsspeichers. Somit könnten Sie in einem Extension Node eine Datenmenge von 4 TB bis 8 TB speichern, wenn dieser über 4 TB Arbeitsspeicher verfügt.

Performance

Die Geschwindigkeit, mit der Daten auf dem Extension Node verarbeitet werden können, hängt maßgeblich vom Overload-Faktor ab. Denn je mehr Arbeitsspeicher als Puffer für den Datenbestand zur Verfügung steht, desto schneller können die Daten verarbeitet werden. Geschwindigkeit ist zwar wichtig, jedoch für den Extension Node nicht ausschlaggebend, denn es handelt sich bei den hier gespeicherten Daten um warme Datensätze. Nach Aussage von SAP schlägt das Datentemperatur-Management mit SAP HANA Extension Node die Performance des Datentemperatur-Managements mit SAP HANA Dynamic Tiering. Genaue Zahlen können Sie bei SAP erfragen, um zwischen dem Einsatz von SAP HANA Extension Node und SAP HANA Dynamic Tiering abzuwägen.

Einsatzszenarien

Als Einsatzszenarien für SAP HANA Extension Node werden zurzeit SAP HANA als native Datenbank, SAP BW auf SAP HANA und SAP BW/4HANA unterstützt. Leider ist ein Einsatz mit SAP S/4HANA oder SAP ERP derzeit noch ausgeschlossen. Für den Einsatz ist mindestens SAP HANA 1.0 SPS12 oder SAP HANA 2.0 erforderlich. Beide Versionen sind seit vielen Jahren auf dem Markt. Sie werden also in Ihrem Alltag kein Problem mit dieser Voraussetzung haben. SAP HANA Dynamic Tiering und SAP HANA Extension Node können in einem SAP-HANA-Verbund auch koexistieren.

Unterstützte SAP-HANA-Standards

SAP HANA Extension Node unterstützt viele Kernfunktionen einer SAP-HANA-Datenbank. Dazu zählen Backup- und Recovery-Funktionen. Sie können beim Einsatz von SAP HANA Extension Node also sicher sein, dass die richtigen Daten auf dem entsprechenden Server wiederhergestellt werden können. Auch SAP HANA System Replication wird unterstützt. Daneben kann SAP HANA Extension Node auch mit Stand-by-Knoten umgehen, wie in Abbildung 2.18 dargestellt ist.

Extension Node aufbauen

Das Umbauen einer SAP-HANA-Datenbank-Maschine in einen Extension Node erfordert einigen administrativen Aufwand. Dazu hat SAP die Hin-

weise 2741690 und 2415279 bereitgestellt. Sie sollten wissen, dass eine Maschine, die zu einem Extension Node umgebaut wurde, keineswegs für immer ein Extension Node bleiben muss. Man kann die Maschine ohne Datenverlust, wenn auch mit etwas Aufwand, wieder zu einer regulären SAP-HANA-Datenbank »zurückbauen«. Dabei müssen Sie darauf achten, dass genügend Arbeitsspeicher zur Verfügung steht, denn, wie bereits erwähnt, kann ein Extension Node zwei- bis viermal so viele Daten speichern wie der Arbeitsspeicher einer normalen SAP-HANA-Datenbank. Die Datenmengen aus dem Extension Node müssen bei einem Umbau von den normalen SAP-HANA-Datenbankinstanzen des Verbunds aufgefangen werden. Alternativ kann man nicht mehr benötigte Daten löschen, was jedoch abhängig von deren Bedeutung nicht immer möglich ist.

2.5 Tabellenpartitionierung

Die SAP-HANA-Datenbank unterstützt eine Tabellenpartitionierung für spaltenbasierte Tabellen. Dabei werden die Datensätze in mehrere Partitionen unterteilt. Die Tabellenpartitionierung können Sie für unterschiedliche Einsatzszenarien nutzen.

Zeilenbegrenzung umgehen

Ein Problem, das Sie mit einer Tabellenpartitionierung umgehen oder beheben können, ist die Zeilenbegrenzung von spaltenbasierten Tabellen. Wenn Ihnen der Alert 27, »Record count of column-store table partitions«, angezeigt wird, bedeutet dass, dass versucht wurde, einen Datensatz, also eine neue Zeile, in eine bereits volle Tabelle einzufügen. SAP-HANA-Datenbanktabellen haben eine Begrenzung von $2^{(32-1)}$, also ca. 2 Milliarden Zeilen innerhalb einer Tabellenpartition. Dieses Limit gilt auch, wenn die Tabelle aus einer einzigen Partition besteht. Wird versucht, mehr als diese Anzahl an Datensätzen einzufügen, kommt es zu dem genannten Fehler. Tritt dieser Fehler in Ihrer Datenbank auf, müssen Sie die Tabelle partitionieren oder umpartitionieren, wenn Sie keine Daten löschen oder aggregieren wollen.

Load Balancing

Ein weiteres Argument für die Tabellenpartitionierung, das Sie bedenken sollten, wenn Sie Tabellen gestalten, ist der Lastausgleich (*Load Balancing*). In einer verteilten SAP-HANA-Landschaft können Daten z. B. an bestimmte Regionen gebunden sein. Wenn man nun eine Tabelle mit Verkaufsdaten anlegt, kann es z. B. sinnvoll sein, jeder Region, in der Absätze des Unternehmens generiert werden, einen eigenen Host zuzuweisen. Auf diesem können Sie dann dank der Partitionierung nur die Verkaufsdatensätze ablegen, die aus der jeweiligen Region stammen. Somit können Sie eine schnelle

regionale Datenverarbeitung erreichen und dennoch von jedem Datenbank-Host aus auch weltweite Abfragen ausführen lassen.

Threading

Tabellenpartitionierungen eignen sich für Systeme mit vielen Prozessorkernen, denn bei einer SQL-Anweisung wird ein *Thread* für jede Tabellenpartition verwendet. In diesem Zusammenhang müssen Sie allerdings beachten, dass auch die Anzahl an Threads begrenzt ist, nämlich durch die maximale Anzahl an zur Verfügung stehenden Prozessorkernen. Zusätzlich kann die Anzahl der Threads über die SAP-HANA-Datenbankparameter eingeschränkt werden. Welche Parameter Sie dazu konfigurieren müssen, beschreibt der SAP-Hinweis 2222250. Sollen mehr Tabellenpartitionen verarbeitet werden, als Threads maximal möglich sind, beeinträchtigt dies die Verarbeitungsgeschwindigkeit. Deshalb sollten Sie niemals zu viele Tabellenpartitionen anlegen.

Delta Merges

Weitere Vorteile aus der Tabellenpartitionierung ergeben sich für Delta Merges (siehe auch Abschnitt 2.2.3). Denn durch die Partitionierung müssen diese nicht mehr für die gesamte Tabelle erfolgen, sondern können stattdessen auf Ebene der Partitionen durchgeführt werden. Datenbankabfragen werden in SAP HANA grundsätzlich vor der Ausführung analysiert. Ein Teil dieser Analyse besteht darin, herauszufinden, wo die zur Durchführung benötigten Datensätze liegen. Ist eine Tabelle partitioniert, wird nur auf die Partitionen zugegriffen, die die benötigten Datensätze beinhalten.

Partitionierung zur Laufzeit

Tabellen müssen nicht von Anfang an partitioniert sein, um die Partitionierung nutzen zu können. Sie können eine nicht partitionierte Tabelle auch nachträglich noch in eine partitionierte Tabelle umwandeln. Dies funktioniert auch in die andere Richtung. Zu beachten ist allerdings, dass Sie bei Tabellen, die über mehrere Host-Maschinen verteilt sind, immer angeben müssen, auf welcher Host-Maschine die nicht partitionierte Tabelle am Schluss der Transformation liegen soll.

Partitionen auflösen

Eine Auflösung von Partitionen erfolgt immer mit dem SQL-Befehl `ALTER` und niemals mit dem Befehl `DROP PARTITION`. Wenn Sie den Befehl `DROP PARTITION` ausführen, erleiden Sie einen vollständigen Datenverlust, denn damit werden die Daten aller Partitionen der betreffenden Tabelle gelöscht. Diese Anweisung verhält sich also genau wie die Anweisung `DROP TABLE`, bei der die gesamte Tabelle gelöscht wird.

Partitionsstrategien

SAP HANA unterstützt drei Strategien, um Tabellen zu partitionieren:

- Hash-Partitionierung
- Round-Robin-Partitionierung
- Range-Partitionierung

Diese Strategien erläutern wir im Folgenden. Dazu verwenden wir den Datensatz aus Tabelle 2.12 als Beispiel.

Artikelnummer	Einkaufspreis (EK)	Verkaufspreis (VK)
1	10	15
2	20	25
3	30	35
4	40	45

Tabelle 2.12 Beispieldatensatz für die Anwendung der Partitionsstrategien

Hash-Partitionierung

Bei der *Hash-Partitionierung* werden Hash-Werte für die Datensätze generiert. Ein Hash-Wert ist die Ausgabe einer Hash-Funktion. Hash-Funktionen werden verwendet, um aus beliebig langen Zeichenketten eine neue Zeichenkette mit fester Länge zu berechnen. Dazu müssen Sie angeben, welche Spalten zur Generierung der Hash-Werte verwendet werden sollen. Sie müssen mindestens eine Spalte auswählen. Abhängig vom Hash-Wert eines Datensatzes wird dieser dann in eine Partition geschrieben. Es ist zudem möglich, Partitionen auf andere Hosts zu schreiben. Da die Datensätze gut balanciert sind, eignet sich die Hash-Partitionierung auch für das Load Balancing.

Mit der SQL-Anweisung in Listing 2.4 können Sie eine hash-partitionierte Tabelle mit zwei Partitionen aus dem Beispieldatensatz aus Tabelle 2.12 erstellen. Die Datensätze werden dabei auf zwei Partitionen und zwei Hosts verteilt.

```
CREATE COLUMN TABLE Beispieltabelle (
  Artikelnummer INT, Einkaufspreis INT, Verkaufspreis INT
                                        )
PARTITION BY HASH (Artikelnummer, Einkaufspreis)
PARTITIONS 2
AT LOCATION '<Host-Name>:30001', '<Host-Name>:30003';
```

Listing 2.4 Erzeugen einer hash-partitionierten Tabelle

Round-Robin-Partitionierung

Eine *Round-Robin-Partitionierung* ist der Hash-Partitionierung sehr ähnlich. Sie kommt jedoch ohne einen Hash-Wert und damit auch ohne die Angabe von Tabellenspalten aus. Die Verteilung der Datensätze auf die Partitionen erfolgt einfach reihum. Dadurch befindet sich jeder Datensatz in Partition `<Datensatznummer> mod <Anzahl an Partitionen>`.

Grundsätzlich sollten Sie die Hash-Partitionierung der Round-Robin-Partitionierung vorziehen, denn ein Partition Pruning ist bei Round-Robin nicht möglich, weshalb alle Partitionen zum Ausführen von Datenbankanfragen in Betracht gezogen werden müssen. Sie können auch beim Round-Robin-Verfahren die Partitionen auf unterschiedliche Hosts verteilen.

Eine Testtabelle für unseren Beispieldatensatz können Sie mit der SQL-Anweisung in Listing 2.5 erstellen:

```
CREATE COLUMN TABLE Beispieltabelle (
  Artikelnummer INT, Einkaufspreis INT, Verkaufspreis INT
                                        )
PARTITION BY ROUNDROBIN
PARTITIONS 2
AT LOCATION '<Host-Name>:30001', '<Host-Name>:30003';
```

Listing 2.5 Erzeugen einer round-robin-partitionierten Tabelle

Abbildung 2.19 zeigt die Verteilung der Beispieldatensätze in einer Round-Robin-Partitionierung mit zwei Host-Maschinen.

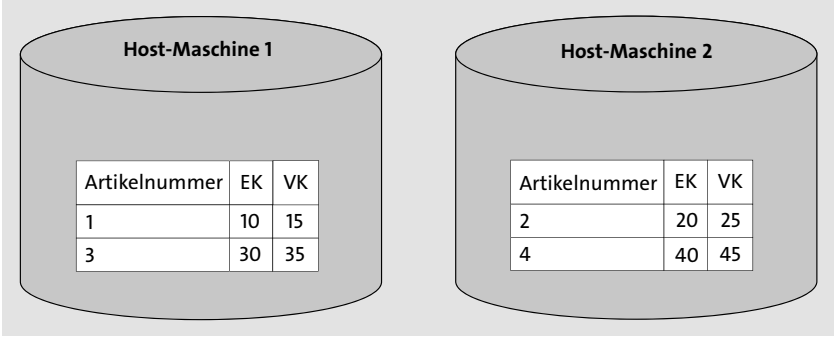

Abbildung 2.19 Round-Robin-Verteilung des Beispieldatensatzes

Range-Partitionierung

Bei einer *Range-Partitionierung* übernimmt nicht der Algorithmus die Verteilung der Datensätze, sondern Sie. Sie müssen bei range-partitionierten Tabellen angeben, nach welcher Spalte die Daten sortiert werden sollen. Zusätzlich müssen Sie die Grenzwerte der einzelnen Partitionen festlegen. Die erzeugten Partitionen lassen sich in Multi-Host-Landschaften auf andere Host-Maschinen verlagern, genauso wie das beim Hash- oder Round-Robin-Verfahren funktioniert.

Um eine eigene Testtabelle auf Basis unseres Beispieldatensatzes zu erstellen, können Sie die SQL-Anweisung aus Listing 2.6 verwenden:

```
CREATE COLUMN TABLE Beispieltabelle (
  Artikelnummer INT, Einkaufspreis INT, Verkaufspreis INT,
  PRIMARY KEY (Artikelnummer, Einkaufspreis)
                                    )
PARTITION BY RANGE (Artikelnummer)(
  PARTITION 1 <= VALUES < 3, PARTITION OTHERS
                                   )
AT LOCATION '<Host-Name>:30001', '<Host-Name>:30003';
```

Listing 2.6 Erzeugung einer range-partitionierten Tabelle

Abbildung 2.20 zeigt die Verteilung des Testdatensatzes mittels Range-Partitionierung auf zwei Host-Maschinen.

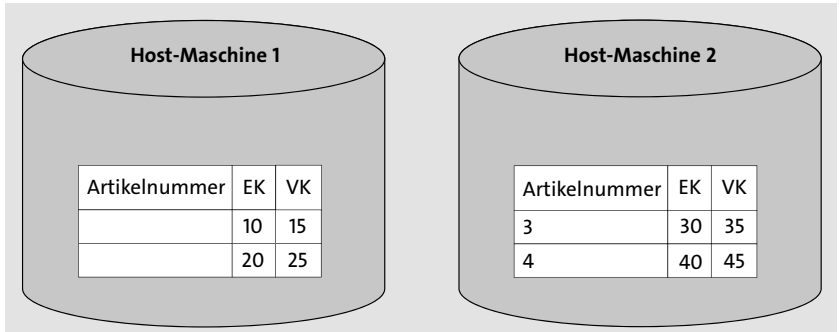

Abbildung 2.20 Range-Verteilung des Beispieldatensatzes

Verteilungsmöglichkeiten zur Laufzeit

Für die Verteilung der Partitionen auf Maschinen ist es egal, welchen Partitionierungstyp Sie einsetzen. Partitionen lassen sich beliebig über die Hosts des SAP-HANA-Verbunds verteilen. Auf einem Host können also gar keine Partitionen einer Tabelle liegen, auf einem anderen einige oder sogar alle Partitionen. Die Partitionen lassen sich mit folgender Anweisung auf andere Host-Maschinen verschieben:

```
ALTER TABLE Beispieltabelle
MOVE PART <Partitionsnummer 1-n>
TO LOCATION '<Host-Name>:30003'
```

Auch nachträglich können Sie die Partitionen noch in ihrer Größe anpassen, indem Sie die Grenzwerte verändern. Somit ist die Partitionierung nicht als *Designtime-Prozess* zu sehen, sondern kann im laufenden Betrieb, also während der *Runtime*, verändert werden. Solche Änderungen können mitunter allerdings großen Einfluss auf die Performance haben und sollten deshalb nicht in Zeiträumen mit großer Last erfolgen.

Single- und Multi-Level-Partitionierung

Neben den verschiedenen Partitionierungsstrategien unterscheidet man zwischen *Single-Level-Partitionierung* und *Multi-Level-Partitionierung*. Single-Level-Partitionierung bedeutet, dass nur *eine* Strategie eingesetzt wird, um die Partitionen zu erstellen. Bei der Multi-Level-Partitionierung werden zwei Strategien gleichzeitig angewandt. Die von SAP HANA 2.0 SPS05 unterstützten Multi-Level-Strategien sind *Hash-Range-*, *Round-Robin-Range-*, *Hash-Hash-* und *Range-Range-Partitionierung*.

Weiterführende Informationen zur Tabellenpartitionierung

Wenn Sie sich umfassender mit der Tabellenpartitionierung auseinandersetzen wollen, sollten Sie den Abschnitt »Table Partitioning« im »SAP HANA Administration Guide for SAP HANA Platform« lesen (siehe *http://s-prs.de/v834117*). Darin wird das Thema sehr ausführlich behandelt und zusätzlich auf weitere Quellen verwiesen.