

IN-MEMORY-DATENBANKEN IM VERGLEICH

Nachdem herkömmliche Datenbanksysteme an Grenzen stoßen, kommen mehr und mehr In-Memory-Lösungen auf den Markt. Lesen Sie, worauf Unternehmen bei der Implementierung achten sollten.

Von Jan Ott und Roland Stirnimann*

Noch stehen In-Memory-Datenbanken (IMDB) vielfach im Schatten der großen klassischen Datenbanklösungen wie der „Oracle Database“, IBMs „DB2“ oder Microsofts „SQL Server“, die sich in den zurückliegenden Jahrzehnten fest in den Unternehmen etabliert haben. Doch im Zuge steigender Anforderungen auf Seiten der Anwender kommt Bewegung in den Markt. Die Anbieter arbeiten mit Hochdruck an neuen Datenbankmodellen und -architekturen wie beispielsweise In-Memory-Lösungen. Glaubt man den Werbeslogans der Softwarehersteller, bietet In Memory eine Reihe von Vorteilen wie beispielsweise eine beschleunigte Datenverarbeitung und damit stark verkürzte Antwortzeiten.

Auf einen Blick

- In-Memory-Technik bildet einen Ansatz, um steigende Anforderungen im Datenbankumfeld zu erfüllen.
- Die Verlagerung der Datenbank in den Hauptspeicher beschleunigt die Verarbeitung der Anfragen und verkürzt die Antwortzeiten.
- Um von den Performance-Vorteilen durch In Memory zu profitieren, müssen jedoch die Applikationen an die neue Technik angepasst werden.
- Die am Markt verfügbaren In-Memory-Lösungen adressieren unterschiedliche Marktsegmente und unterscheiden sich im Funktionsumfang sowie den Möglichkeiten zur Integration und Administration.

Allerdings haben Anwender, die überwiegend in der Welt relationaler Datenbanksysteme verhaftet sind, noch viele Fragen. Was ist beim Praxiseinsatz von In Memory zu beachten? Ist die neue Technologie in jedem Fall schneller, macht sie andere Datenbanken sogar überflüssig? Diese Fragen gilt es für die Unternehmen zu klären, um die richtige Technik für die eigenen Anforderungen zu finden.

Mittlerweile gibt es zahlreiche In-Memory-Datenbanken auf dem Markt. Vier Produkte sollen nachfolgend näher beschrieben werden. Je zwei davon bieten umfangreiche Features für den Einsatz im Enterprise-Segment („Oracle TimesTen“ und „SolidDB“ von IBM) beziehungsweise eignen sich eher für den Betrieb in klein oder mittelgroß dimensionierten IT-Umgebungen („HyperSQL Database“ und „SQLite“).

Oracle TimesTen

Oracle hat TimesTen im Jahr 2005 gekauft und die gleichnamige In-Memory-DB seitdem kontinuierlich weiterentwickelt. Hinsichtlich Funktionsumfang und Lizenzierung bewegt sich die Datenbanklösung auf Enterprise-Niveau und ist sowohl für Linux und Unix als auch für Windows-Plattformen erhältlich. Neben dem Betrieb als eigenständige Datenbank lässt sich TimesTen vor allem auch als Cache für bestehende Oracle-Tabellen verwenden. Diese werden dabei vollständig oder partiell als sogenannte Cache Groups in TimesTen geladen. Spezielle Prozesse sorgen anschließend dafür, dass die Daten – je nach Anwendungsfall

– synchron oder asynchron in die Oracle-Datenbank geschrieben werden. Das beschleunigt den Zugriff auf Business-relevante Geschäftsdaten und sorgt auch bei hoher Systemlast für einen unterbrechungsfreien Betrieb der zugehörigen Datenbank-Server. Auch das automatische Nachladen neuer Daten aus dem relationalen Datenbank-Management-System (RDBMS) funktioniert problemlos, leider wird jedoch für die Cache Groups bisher nur die Oracle-Plattform unterstützt.



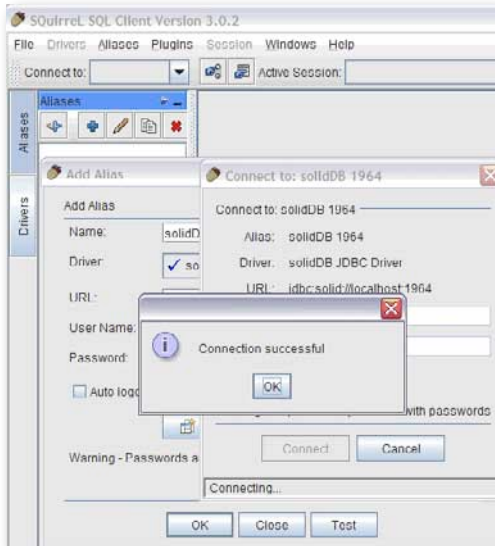
Damit der In-Memory-Datenbank nicht der Platz ausgeht, muss der Cache regelmäßig bereinigt werden (Aging). TimesTen bietet dazu zwei Vorgehensweisen – „time-based“ oder „least recently used“ (LRU). Dabei passt sich die Lösung immer mehr an das Oracle RDBMS an. So findet der Datenbankadministrator zum Beispiel ein Data Dictionary mit Tabellen wie DBA_USERS oder DBA_OBJECTS vor. Datenbankentwickler können sich zudem über die Unterstützung von PL/SQL freuen.

Dennoch finden sich bei Weitem noch nicht alle Pakete und SQL-Funktionen aus dem Oracle RDBMS auch in TimesTen. Beim Portieren einer Applikation vom Oracle RDBMS auf TimesTen sind viele Anpassungen am Code erforderlich. Den damit verbundenen Zeitaufwand sollten Anwender nicht unterschätzen und in ihrer Budget- und Projektplanung berücksichtigen.

TimesTen verfügt zudem über diverse Replizierungsvarianten (synchron/asynchron, uni-/bidirektional). So lässt sich beim Ausfall einer TimesTen-Instanz sofort ein Failover initiieren und somit ein unterbrechungsfreier Betrieb sicherstellen.

SolidDB von IBM

Die In-Memory-Lösung wurde 2007 von IBM aufgekauft und seither weiterentwickelt. Die aktuelle Version 6.5 von SolidDB ist für die IBM-eigene Power7-Prozessorgeneration optimiert. Neben IBMs Unix-Derivat AIX läuft die In-Memory-Datenbank auch auf anderen Plattformen wie Linux, Win-



dows und Solaris. Verwalten lässt sich SolidDB über die gleiche SQL-Schnittstelle wie die Datenbanklösung IBM DB2. In Bezug auf Funktionsumfang und Einsatzgebiet ähnelt SolidDB stark der Oracle-Lösung TimesTen. Gegenüber Entwicklern verhält sie sich jedoch ähnlich einer DB2-Instanz. So kann „SolidDB Universal Cache“ ebenfalls als Cache von Tabellen verwendet werden, die aus einer klassischen relationalen Datenbank stammen. Im Gegensatz zu TimesTen unterstützt SolidDB allerdings mehr Plattformen. Die Tabellen können aus DB2-, Microsoft-SQL-Server-, Informix-, Sybase- oder Oracle-Datenbanken stammen.

In Sachen Hochverfügbarkeit bietet SolidDB die Replikation zu einer weiteren Instanz sowie ein entsprechendes „Session

Die Datenbankmodelle im Vergleich

In Memory	On Disk
Applikation und In Memory auf gleichem Server, verbunden mit „Direct Connect“.	Verschiedene Server für Applikation und Datenbank.
Keine blockierenden Disk-I/O-Operationen beim Schreibvorgang in die Datenbank (Commit), da Verarbeitung im Hauptspeicher.	Blockierende I/O-Operation beim Commit, da der Log Buffer auf die HDD geschrieben wird.
Verlust von „Committed“-Daten möglich, da nur im Hauptspeicher (Log Buffer).	AKID (Atomarität, Konsistenz, Isoliertheit und Dauerhaftigkeit) garantiert.
Wenig Backup-Features, Datenverlust beim Server-Crash möglich.	Umfangreiche Backup-Features, Restore/Recover ohne Datenverlust.
Wenige Analyse-Tools bei Performance-Problemen (keine SQL Elapsed Time).	Umfangreiche Tools und Reporting-Möglichkeiten.
Keine parallelen Abfragen (Parallel Query).	Parallel Query vorhanden/möglich.
Bisher wenig Wissen vorhanden.	Viel Wissen vorhanden.

Quelle: Trivadis

Failover“, falls die aktive Instanz unerwartet beendet wird. Das könnte beispielsweise dann der Fall sein, wenn die Datenbank abgestürzt ist. Anwender können darüber hinaus in SolidDB so genannte Aging-Mechanismen nutzen, um alte oder nicht mehr verwendete Daten schnell und zuverlässig aus der In-Memory-Datenbank zu entfernen.

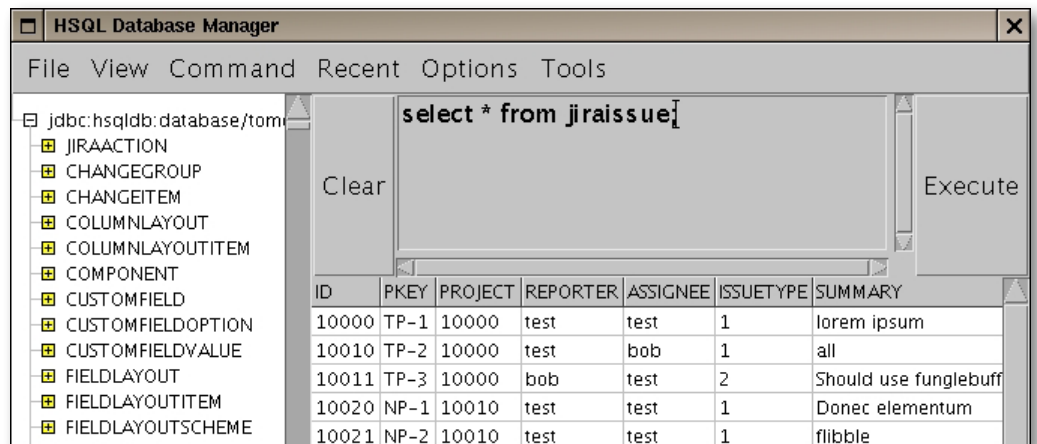
HSQldb – HyperSQL Database

Die komplett in Java geschriebene In-Memory-Datenbank HSQldb steht unter der BSD-Lizenz und ist sowohl im Open-Source-Umfeld als auch im kommerziellen Bereich weit verbreitet. Die Lösung beansprucht als komplette Installation nur 1,3 Megabyte Platz, funktioniert auf allen Java-fähigen Betriebssystemen und ist komplett kostenlos. Aufgrund des offen liegenden Sourcecodes gibt es auf dem Markt bereits eine Vielzahl unterschiedlicher Anbieter. HSQldb wird für viele verschiedene Applikationen wie beispielsweise „OpenOffice“, „Mathematica“ und „InstallAnywhere“ als Persistenz-Layer eingesetzt.

Als Einsatzgebiet ist der Bereich der kleinen bis mittelgroßen In-Memory-Datenbanken vorgesehen. Die Lösung kann sich hinsichtlich des Funktionsumfangs bei Weitem nicht mit Enterprise-Lösungen wie TimesTen oder SolidDB messen. Die Tabellen in HSQldb lassen sich wahlweise als reine In-Memory-Tabellen beziehungsweise als Disk-basierende oder persistente Tabellen anlegen. Bei Letzteren werden zusätzlich drei weitere Typen unterschieden:

- **MEMORY (default):** Daten/Objekte sind in Form eines SQL-Scripts auf der Festplatte gespeichert.
- **CACHED:** Binäres Dateiformat auf der Festplatte. Nicht alle Daten sind im Memory geladen.
- **TEXT:** Tabellendaten sind in CSV-Dateien gespeichert.

Jeder der genannten Disk-basierenden Typen garantiert bei einem Server-Absturz die Persistenz der „committed“ Transaktionen. HSQldb lässt sich entweder im so genannten In-Process- oder im „Server-Mode“ betreiben. Während In Process ▶



► schneller arbeitet, bietet der Server Mode mehr Flexibilität, da auch via Netzwerk auf die Datenbank zugegriffen werden kann. Beim In-Process-Mode befinden sich die Applikation und die Datenbank im selben Speicherbereich, so dass keine Zeit durch die Latenz auf dem Netzwerk verloren geht.

Die Verwaltung der Datenbank erfolgt entweder über ein einfaches Kommandozeilen-Programm oder ein Web-basierendes Interface. Besonders zu erwähnen ist bei HSQLDB die gute Dokumentation in vielen Büchern. Dort geht es meist um die Verwendung der Datenbank im Zusammenhang mit Frameworks wie „Hibernate“ oder „Spring“. Die Feature-Liste von HSQLDB hebt besonders den guten SQL-Support hervor. Darüber hinaus bietet HSQLDB umfangreiche Java- beziehungsweise JDBC-Unterstützung, da die Datenbanksoftware komplett in Java geschrieben ist.

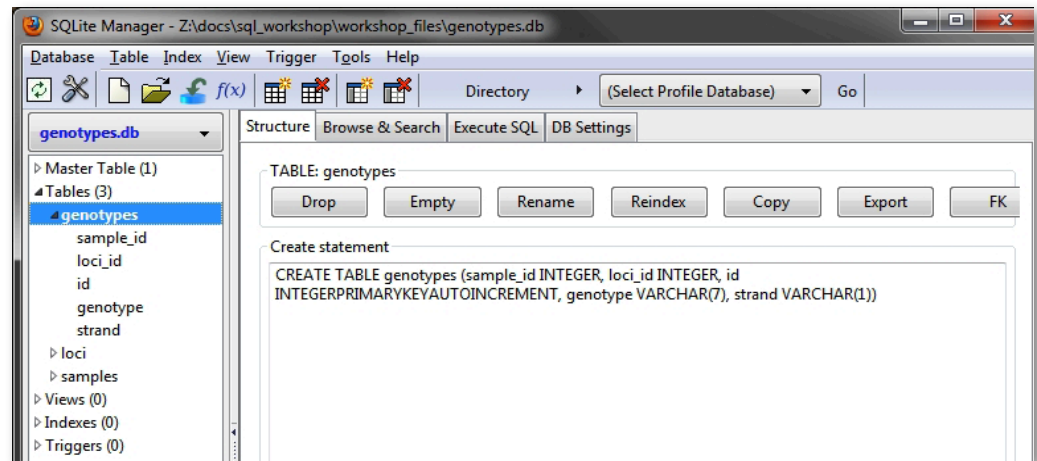
SQLite

Die In-Memory-Datenbank SQLite adressiert wie HSQLDB das Segment kleiner beziehungsweise mittelgroßer Installationen und eignet sich für Applikationen, die Daten in einem Cross-Platform-Format speichern sollen. Die Lösung ist außerdem gut als Embedded-Datenbank für MP3-Player oder PDAs sowie kleine Websites brauchbar. Das Produkt ist unter einer Public-Domain-Lizenz verfügbar und kann von den Anwendern kostenlos aus dem Netz heruntergeladen werden.

Im Gegensatz zu den meisten anderen Produkten erfordert SQLite keinerlei Installation oder Konfiguration. Der Grund: Die Datenbank ist nur wenige hundert Kilobyte groß und verfügt über keinen eigenen Server-Prozess. Die Daten werden von der Applikation direkt aus der Datendatei in den Speicher geladen. Als einzige externe Administrationsmöglichkeit bietet SQLite ein Command-Line-Interface an. Obwohl mehrere Applikationen die gleiche Datendatei verwenden können, ist die Datenbank laut Hersteller nicht auf hohe Parallelität ausgelegt. SQLite wird dennoch als Persistenz-Layer von Firefox, Skype, Apple und Adobe verwendet. Das breite Einsatzspektrum ist ein Beleg dafür, dass sich die Datenbank effektiv in Kombination mit vielen Programmiersprachen verwenden lässt. Neben der Unterstützung aller gängigen Plattformen kann die Datenbank zudem relativ einfach auf weitere Betriebssysteme portiert werden.

Das leisten In-Memory-Datenbanken

Eine In-Memory-Datenbank stellt im Grunde ein besonderes Datenbank-Feature dar, mit dem sich Business-Anwendungen er-



heblich beschleunigen lassen. Sofern für eine Plattform beziehungsweise Programmiersprache ein Treiber existiert, lässt sich eine In-Memory-Lösung grundsätzlich wie ein herkömmliches RDBMS ansprechen. Allerdings ist dafür die betreffende Applikation an die Funktionen der Datenbank anzupassen und nicht umgekehrt. Der bloße Einsatz der In-Memory-Technologie garantiert noch lange keine zuverlässigen Performance-Vorteile. Der Grund: Wie bei vielen anderen Anwendungen gibt es keine funktionierende Out-of-the-Box-Lösung. Vor allem im Enterprise-Segment sind unternehmensspezifische Anpassungen notwendig. Herausforderungen und Probleme liegen oft im Detail und treten meist erst während einer Proof-of-Concept-Phase zutage.

Folgende Anforderungen und Rahmenparameter sprechen häufig für den Einsatz einer In-Memory-Datenbank:

- Hoch frequentierter Lesezugriff auf den Daten-Cache (Lookup-Tabellen);
- temporäre Daten mit vielen Zugriffen (lesen und schreiben);

Einordnung: In-Memory-DBs

Bei der Auswahl der In-Memory-DB spielen die zu erwartende Größe der Datenbank sowie die Anzahl der Transaktionen eine entscheidende Rolle.



Quelle: Trivadis

- Entlastung des Datenbank-Servers (Anzahl Transaktionen, I/O, User Sessions);
- Ausführung einfacher, kurzer SQL-Statements;
- Datenverlust kann in Kauf genommen werden;
- Datenbank kann/soll nahe der Applikation liegen (gleicher Server, Direct Connect).

So funktioniert In Memory

Beim Studium der Dokumentationen zeigt sich, dass sämtliche In-Memory-Datenbanken im Grunde auf ähnlichen Prinzipien basieren. Vor allem bei komplexeren Produkten wie TimesTen oder SolidDB sind viele Gemeinsamkeiten festzustellen.

- 1 Datenhaltung:** Beim Starten der Datenbank lädt das System den gesamten Datenbestand von der Festplatte in den Speicher, damit bei laufendem Betrieb keine Daten nachgeladen werden müssen.
- 2 Checkpoint Files/Snapshot Images:** Geänderte Daten werden in regelmäßigen Abständen mit dem persistenten Speicher (Festplatte) abgeglichen.
- 3 Transaction Logs:** Zwischen den einzelnen Checkpoint-Vorgängen werden laufende Änderungen in Transaction Logs geschrieben, um nach einem Crash ein Rollforward machen zu können.
- 4 AKID-Prinzip:** Wie herkömmliche RDBMS können auch In-Memory-Datenbanken sämtliche Daten nach dem AKID-Prinzip (Atomarität, Konsistenz, Isoliertheit und Dauerhaftigkeit) verarbeiten.
- 5 Hochverfügbarkeit:** In-Memory-Produkte wie IBM SolidDB oder Oracle TimesTen bieten Hochverfügbarkeit in Form von Datenbankreplikation an. Bei einem unerwarteten Crash kann ein Failover auf die verbleibende Instanz erfolgen.
- 6 Direct Connect:** Die Applikation hat die Möglichkeit, die In-Memory-Datenbank direkt in ihren eigenen Adressbereich zu laden, um jeglichen Overhead – wie etwa sämtliche Netzprozesse – zu vermeiden.

Der größte Unterschied zwischen einem Disk-basierenden und einem In-Memory-System besteht in der Datenhaltung. Ein Disk-basiertes RDBMS lädt erst bei Bedarf die erforderlichen „Data Pages“ von der Disk in den „Buffer Pool“ nach, während etwa TimesTen beim Starten der Datenbank alle Daten aus dem Checkpoint-File in den Speicher liest. Dadurch vereinfachen sich die Zugriffsalgorithmen bei In-Memory-Systemen entscheidend, was in der Praxis zu einer besseren Anwendungs-Performance führt. Der „Query Optimizer“ von TimesTen kann zudem direkt auf alle Daten im Hauptspeicher zugreifen, ohne zusätzlichen I/O-Traffic zu erzeugen.

In-Memory-Erfahrungen

Aus Projekterfahrungen mit Oracle TimesTen resultieren zwei grundlegende Erkenntnisse:

1 Die Umstellung auf In Memory stellt ein eigenes IT-Projekt dar: Eine bestehende Applikation lässt sich weder schnell und spontan umstellen, noch garantiert In Memory automatisch eine beschleunigte Anwendungs-Performance. Die Migration der Applikationen erfordert von den Anwendern, diese vorher anzupassen.

2 Die Implementierung von In Memory erfordert Know-how und Zeit: Für die Implementierung von In-Memory-Datenbanken sind fundierte Kenntnisse in Datenbanken und Anwendungsintegration erforderlich. Bestehende Business-Applikationen anzupassen und zu testen bedeutet für die Anwenderunternehmen Aufwand, der sich angesichts der neuen Technik oft nur schwer kalkulieren lässt.

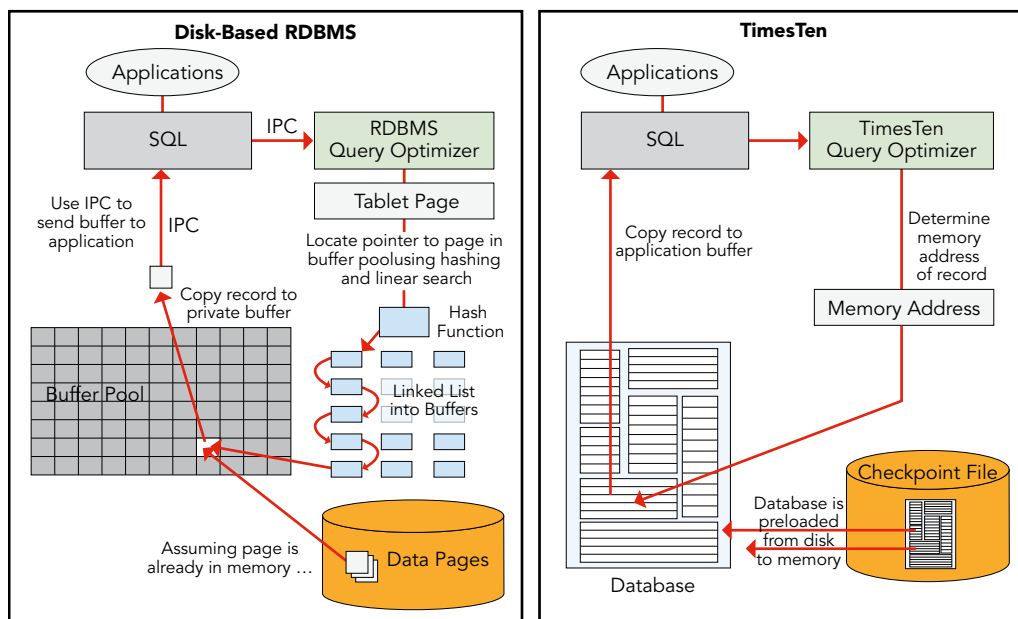
Auf Grundlage langjähriger Erfahrungen mit Oracle-Datenbanksystemen lässt sich feststellen, dass bestimmte Tatsachen aus dem Oracle RDBMS weder implizit für TimesTen noch grundsätzlich für In Memory gelten. Je nach Komplexität eines DB-Projekts reicht reines Oracle-RDBMS-Know-how mitunter nicht aus, da sich In-Memory-Datenbanken in bestimmten Situationen komplett anderes verhalten.

Das gilt beispielsweise für Schreibvorgänge (Commit): TimesTen speichert die Daten im flüchtigen Hauptspeicher, während Oracle RDBMS sie auf Festplatten oder SSDs ablegt. Eine Umstellung auf In-Memory-Technik erfordert daher umfangreiche Änderungen im Programmcode sowie die Implementierung von zusätzlichen Sicherheitsmechanismen gegen Datenverlust. Um die Ergebnisse einer Implementierung zu validieren, ist darüber hinaus eine regelmäßige Performance-Analyse erforderlich.

Die Hauptgründe dafür liegen darin, dass TimesTen bestimmte Programmpakete sowie

Klassische RDBMS versus In-Memory-Datenbank

Während die Disk-basierte RDBMS die Data Pages erst bei Bedarf in den Buffer Pool liest, lädt TimesTen schon beim Start der DB alle Daten in den Speicher.



Quelle: Oracle

Funktionen fehlen und die Anwendungen oft komplexe Datenbankabfragen enthalten. In früheren Projekten mussten beispielsweise die „Connect-by“-Ausdrücke in SQL geändert werden, um den gleichen Output auch ohne sie zu erreichen. Der dafür notwendige Arbeitsaufwand darf keinesfalls unterschätzt werden und stellt selbst erfahrene Entwickler vor Herausforderungen.

Perfekt formatierte Performance Reports, wie sie vom Oracle RDBMS bekannt sind, fehlen in TimesTen ebenfalls. Um „Data Dictionary Views“ zu analysieren, müssen sich Datenbankadministratoren deshalb mit SQL-Queries behelfen. Erschwerend kommt hinzu, dass TimesTen keine „Elapsed Time“ aufzeichnet.

Der Aufwand für die Umstellung bestehender Applikationen auf In Memory ist nicht zu unterschätzen, der Übergang sollte daher als eigenständiges IT-Projekt geplant werden. Entscheider müssen einen großzügigen Zeitpuffer für Softwareanpassung, Testing und Analyse einkalkulieren und entsprechende Ressourcen bereitstellen.

Fazit

Für IT-Entscheider stellt sich die Frage, ob sich ein Umstieg auf In-Memory-Datenbanken lohnt oder ein Upgrade der bestehenden Systeme die bessere Alternative darstellt. Fakt ist, dass In-Memory-Datenbanken in bestimmten Anwendungsszenarien schneller arbeiten als herkömmliche Systeme, zum Beispiel wenn „Direct-Connect“-Verbindungen genutzt werden kön-

nen und relativ einfach strukturierte SQL-Abfragen in großer Menge anfallen. Eine Migration auf eine In-Memory-Lösung verursacht aber auch erhebliche Kosten für zusätzliche Hardware, Softwarelizenzen sowie einen erhöhten Aufwand für Wartung und Backup/Recovery-Mechanismen. Das Hauptaugenmerk der Anwender sollte vor allem darauf liegen, eine möglichst hohe Ausfallsicherheit zu gewährleisten. Denn In-Memory-Systeme erlangen ihren Geschwindigkeitsvorteil vor allem durch die Nutzung des flüchtigen Arbeitsspeichers, wo ein Stromausfall einen sofortigen Datenverlust zur Folge hätte. Produkte wie Oracle TimesTen umgehen dieses Problem, indem sie zusätzlich zu einer vorhandenen On-Disk-Instanz operieren und Datenänderungen auch auf Festplatte sichern.

Die Verantwortlichen in den Anwenderunternehmen müssen sich außerdem überlegen, ob die Migration auf ein In-Memory-System wirtschaftlich sinnvoll ist. In Zeiten sinkender Preise für Hardwarekomponenten ist es heute auch für kleine und mittelständische Unternehmen möglich, entsprechend den In-Memory-Anforderungen leistungsstarke Server-Systeme zu beschaffen. Eine Rolle spielt dabei, ob das nötige Know-how zur Anpassung vorhandener Anwendungen im eigenen Unternehmen vorhanden ist und genügend finanzielle Mittel zur Verfügung stehen. (ba)

*Jan Ott und Roland Stirnimann sind Senior Consultants bei Trivadis.