# Einführung in-memory DB TimesTen

**Carsten Herbe**
**Metafinanz-Informationssysteme GmbH**
**München**

**Schlüsselworte**
TimesTen, in-memory, Architektur, Exalytics

## Einleitung

In-memory Datenbanken sind zurzeit in aller Munde. Oracle bietet mit TimesTen auch eine relationale in-memory Datenbank an, welche nicht nur Hauptbestandteil der Exalytics-Appliance ist, sondern auch als Standalone Lösung auf beliebiger Hardware betrieben werden kann. Waren früher eher OLTP-Systeme Hauptanwendungsszenario, geraten in letzter Zeit auch immer mehr dispositive Systeme - z.B. Datamarts - in den Vordergrund. Durch sinkende Speicherpreise sind heutzutage auch Systeme mit einem Terabyte Hauptspeicher finanzierbar. Dadurch können auch größere Datenmengen in-memory verarbeitet werden.
Oracles in-memory Datenbank TimesTen gibt es jetzt schon seit 1998. Im Laufe der Jahre wurde die Software immer weiterentwickelt. Es kamen neue Features hinzu und der Funktionsumfang wurde immer mehr an die „klassische“ Oracle Datenbank angeglichen. Selbst PL/SQL ist in TimesTen verfügbar, wenn auch mit ein paar Einschränkungen.
In diesem Paper wird die Architektur und Funktionsweise von TimesTen vorgestellt. Die Verwendung von TimesTen als Cache vor einer Oracle Datenbank wird ebenso beschrieben. Abschließend vergleichen wir den Funktionsumfang (SQL, PL/SQL, Tabellen & Indices, Storageoptionen) der Standalone TimesTen in der Version 11.2 gegenüber der Exalytics und "klassischen" Oracle Datenbank.

## Architektur

Alle Daten wie Tabellen und Indexes werden im Hauptspeicher gehalten und dort verarbeitet. Aber auch eine in-memory Datenbank kommt nicht ganz ohne Festplatte aus. Regelmäßig werden Abbildungen des Speichers in Checkpoint Files geschrieben. Und analog zu den Redo-Logs der Oracle Datenbank gibt es in TimesTen Transaction Logs.
Für eine Applikation gibt es zwei unterschiedliche Wege, sich mit einer TimesTen zu verbinden. Entweder klassisch über das Netzwerk und entsprechende Treiber (ODBC, JDBC, …) oder als *direct linked Application*. Letzter Weg setzt voraus, dass die Applikation auf dem gleichen Server wie TimesTen läuft. Dann kann sie über Shared Libraries direkt auf die Speicherstrukturen von TimesTen zugreifen, d.h. der Overhead entfällt. Dieser Weg ist natürlich für performance-kritische OLTP Anwendungen mit vielen kleinen Transaktionen sehr interessant.

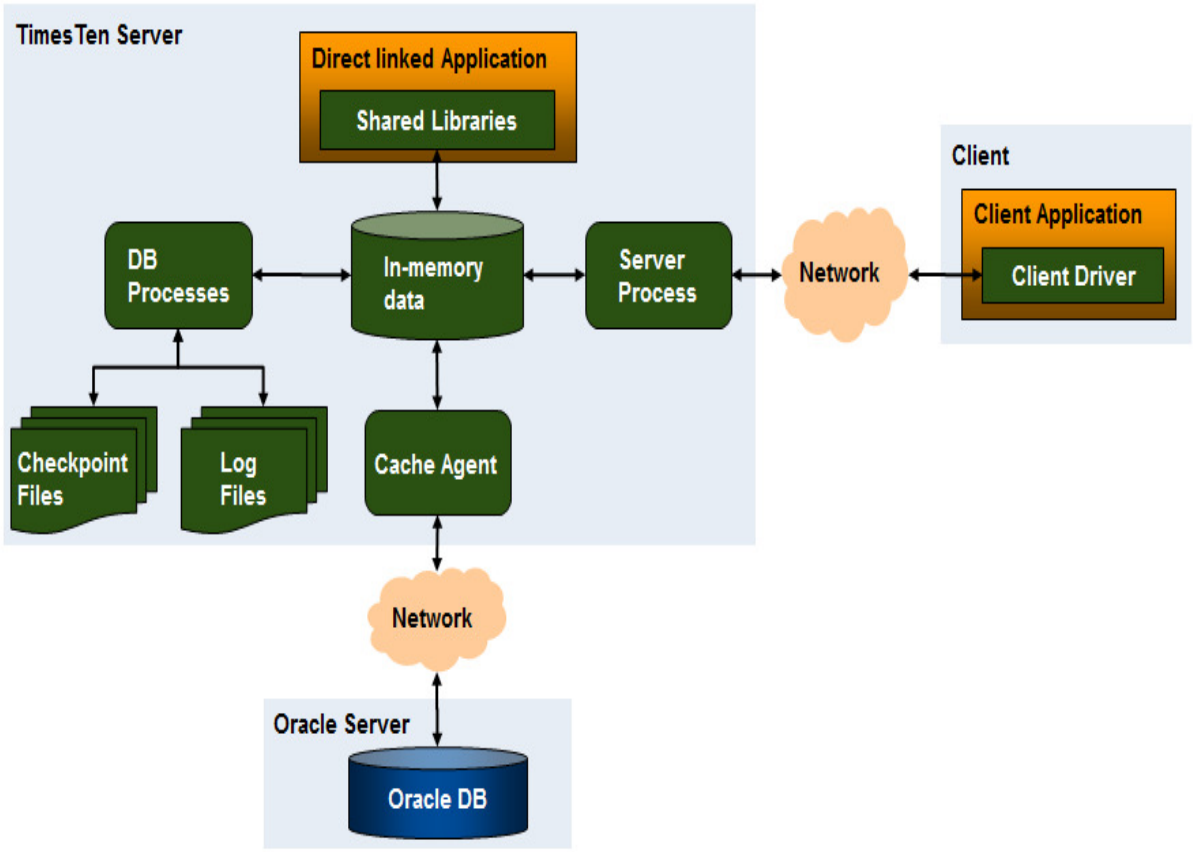


*Abb. 1: TimesTen Architektur*

Man kann TimesTen als eigenständige Datenbank betreiben. Oder man nutzt sie als Cache für eine „klassische“ Oracle Datenbank. Hier kann man Tabellen komplett oder teilweise nach TimesTen laden. Für die Synchronisation der Daten ist der Cache Agent, ein eigener Prozess, verantwortlich. Durch sogenannte Cache Groups, welche aus einer oder mehreren Tabellen mit Foreign Key Beziehungen bestehen, lässt sich sicherstellen, dass Daten über Tabellen hinweg konsistent im Cache landen.

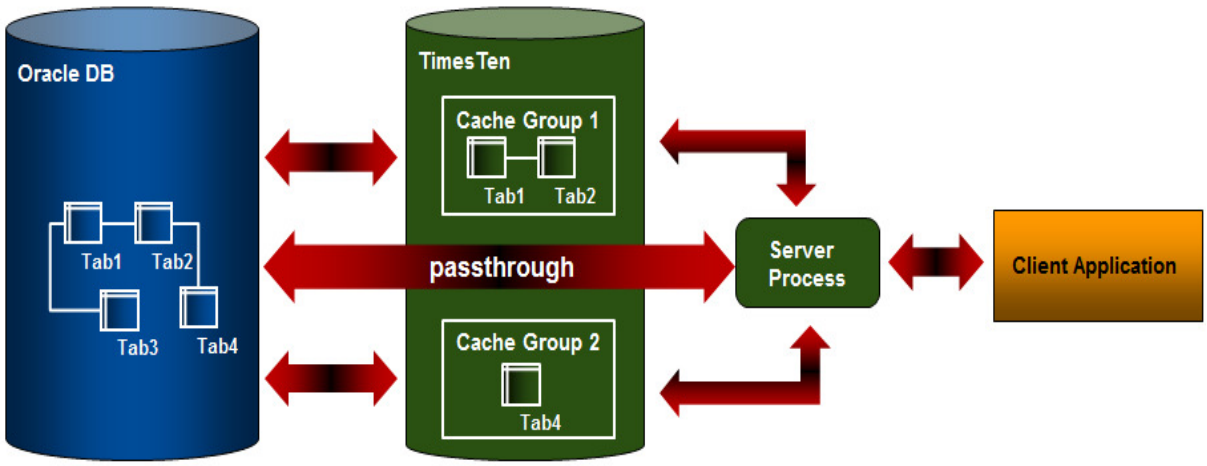


*Abb. 2: Cache Groups & Passthrough*

Kann eine Anfrage nun nicht von TimesTen beantwortet werden, wird sie einfach an die zugrunde liegende Oracle Datenbank weitergereicht.
Um die Cache Funktionalität nutzen zu können, muss auf der Oracle Datenbank aber ein spezieller User angelegt werden, der die Änderungen an den betroffenen Tabellen überwacht.
Tabellen müssen auch nicht komplett in den Cache geladen werden. Die Datenmenge lässt sich über eine WHERE-Clause einschränken. Dies ist besonders interessant, wenn man TimesTen als Datamart einsetzt. So lassen sich die aktuellen Datensätze in der schnellen TimesTen halten und gleichzeitig kann man noch über Passthrough auf die ältere Daten in der Oracle Datenbanken zugreifen.
Eine entscheidende Einschränkung gibt es aber bei der Verwendung von Cache Groups. Die Tabellen müssen einen Primary Key haben. Leider ist dies in Datamarts gerade bei Faktentabellen oft aber nicht der Fall. Das bedeutet, dass man gegebenenfalls die Faktentabellen um einen künstlichen Schlüssel erweitern müsste.

**High-Availability**

Um High-Availability zu gewährleisten, lässt sich per Real-Time Replication eine Hot Standby Datenbank befüllen. Aus dieser lassen sich einfach weitere Subscriber-Datenbanken befüllen, auf welche ebenfalls rein lesend zugegriffen werden kann.

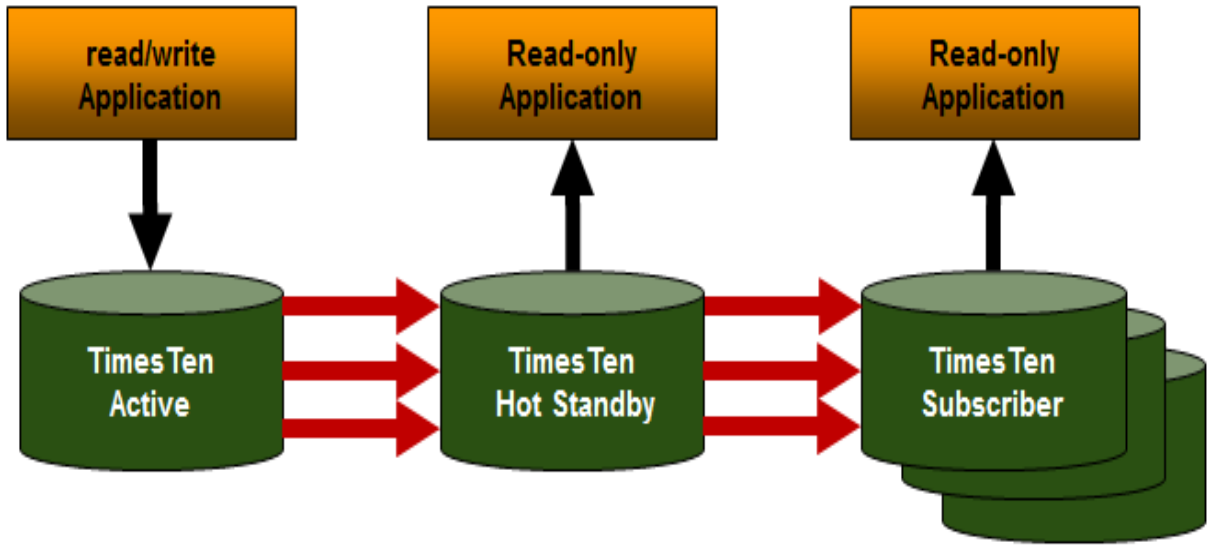


*Abb. 3: Real-Time Replication*

**TimesTen & Exalytics**

TimesTen ist auch integraler Bestandteil der Exalytics Appliance von Oracle. Exalytics dient als in-memory Cache für verschiedene Datenquellen (u.a. natürlich Oracle) und bringt mit einer angepassten OBIEE Version auch gleich das passende BI Tool mit. Der Server nutzt TimesTen (und auch Essbase OLAP) um die Daten in-memory zu halten. Die Cache-Funktionalität von TimesTen wird hier nicht genutzt. Die hier verwendete *TimesTen for Exalytics* ist eine speziell angepasste Version von TimesTen. Die Unterschiede zwischen den beiden Versionen werden am Ende des Papers beschrieben.

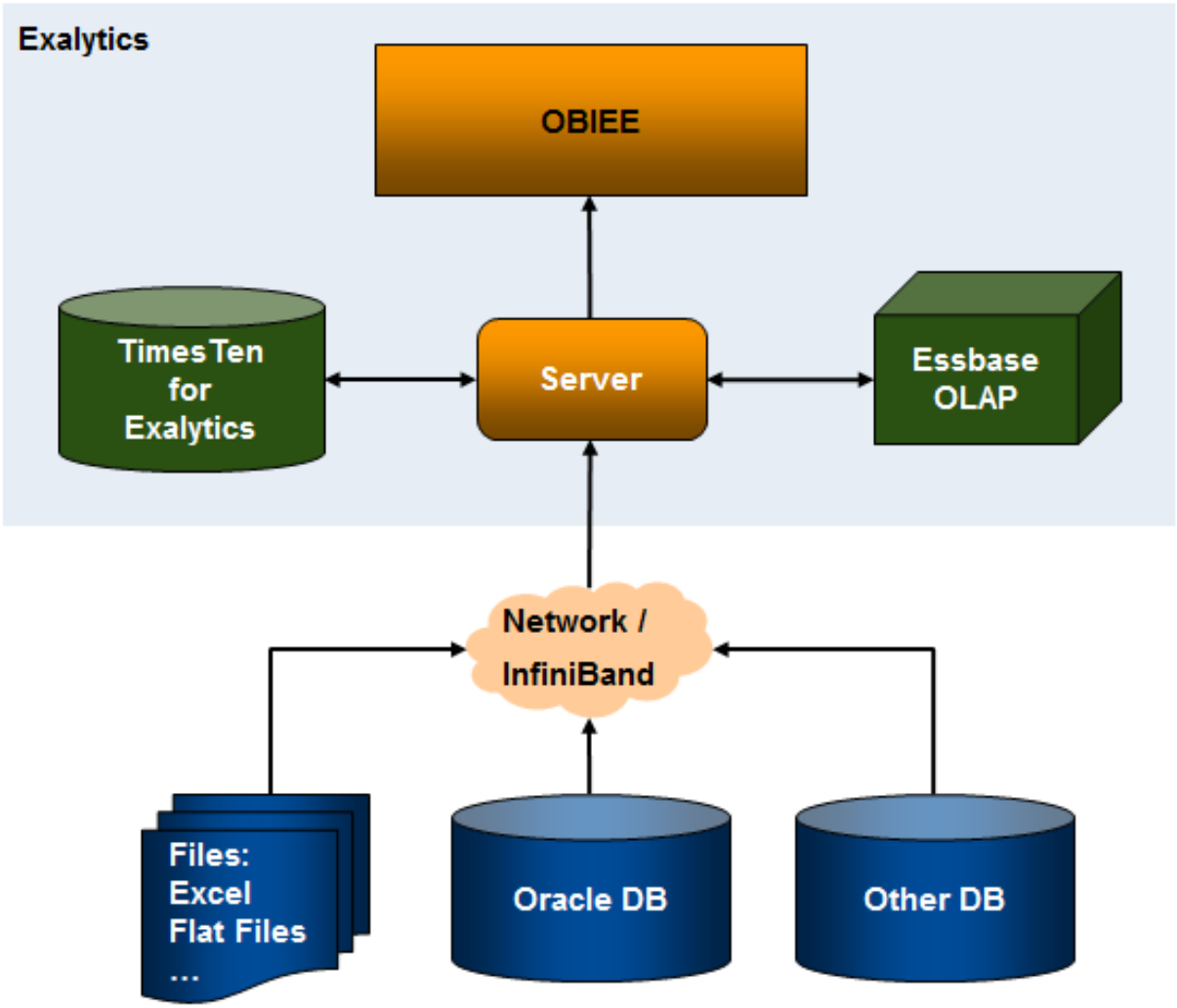


*Abb. 4: Architektur Exalytics*

**TimesTen als Cache**

In diesem Abschnitt wird vorgestellt, wie man mit mehreren TimesTen Instanzen horizontal skalieren kann und welche Synchronisationsmöglichkeiten es zwischen den Cache Groups und der Oracle Datenbank gibt.

Mehrere TimesTen Instanzen auf verschiedenen Servern lassen sich zu einem Cache Grid zusammenfassen. Eine solche Instanz wird als Grid Member bezeichnet.

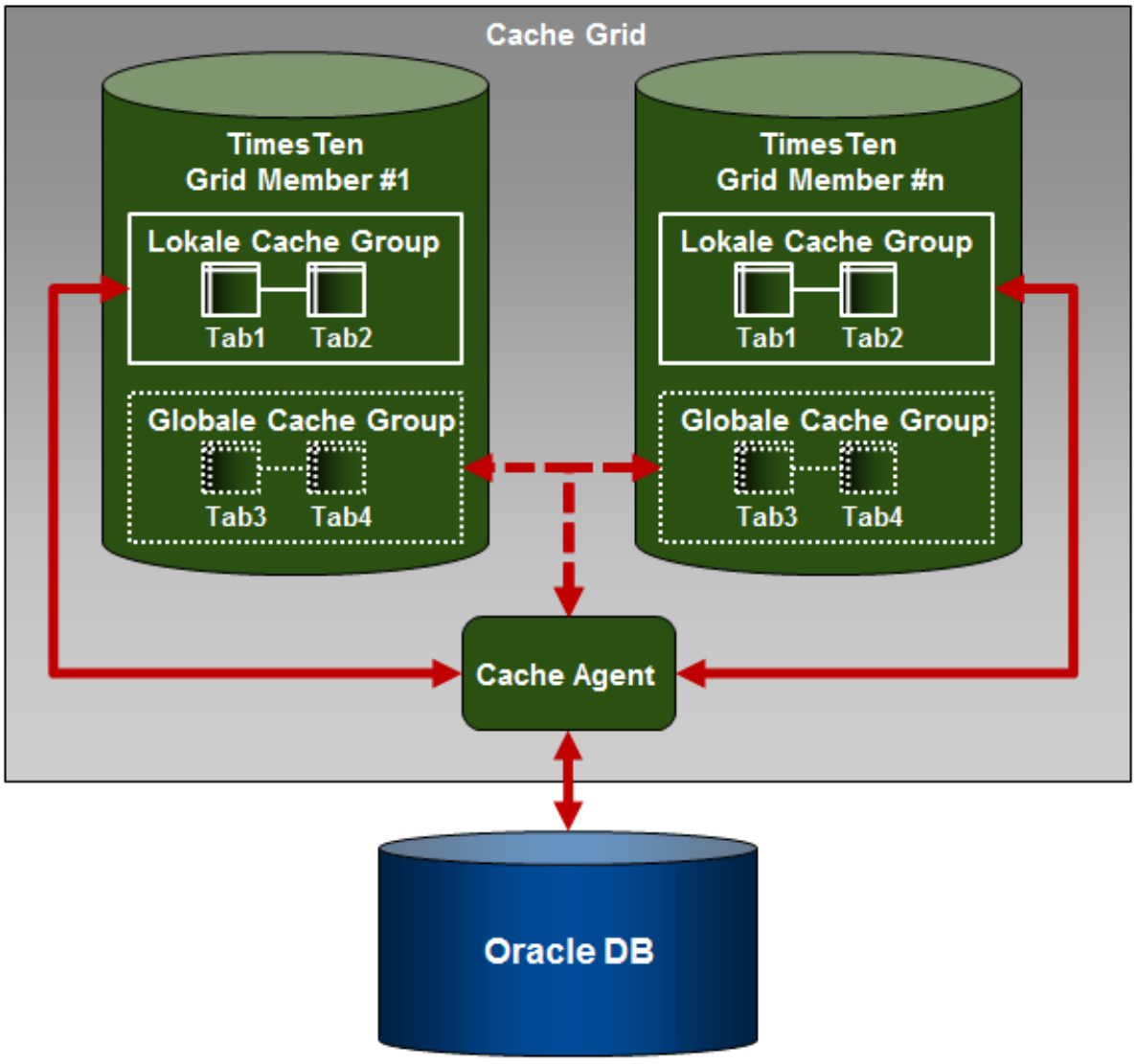


*Abb. 5: Cache Grid*

In einem Cache Grid kann als lokale und globale  Cache geben. Die Daten der lokalen Cache Groups werden in jedem Grid Member gespeichert, für das die Cache Group definiert ist. Dies bietet sich z.B. für Tabellen mir Referenzdaten an, welche oft gejoint werden und eher statisch sind. Es ist bei Updates auf Daten einer lokalen Cache Group nicht sichergestellt, in

welcher Reihenfolge die anderen Members aktualisiert werden und es kann zu Inkonsistenzen kommen.
Die Daten aus Globalen Cache Groups dagegen werden von allen Grid Members geteilt, d.h. sie werden in das Grid Member verschoben, in dem sie gerade benötigt werden. Dies funktioniert ähnlich zu Cache Fusion des Oracle RACs. Da die Daten nur einmal im Grid verfügbar sind, sind Updates damit automatisch immer konsistent.
Abfragen über eine Tabelle lassen sich global über sämtliche Grid Members ausführen. Werden andere Tabellen gejoint, so geschieht dies immer nur lokal.
Für die Synchronisation zwischen den Cache Groups und der Oracle Datenbank gibt es verschieden Konfigurationsmöglichkeiten:

- Read-only Cache Group
- Asynchronous Writethrough (AWT) Cache Group
- Synchronous Writethrough (SWT) Cache Group
- User managed Cache Group

Wie der Name vermuten lässt, werden in der Read-only Cache Group Daten in TimesTen nur gelesen. Sollten die Daten nicht verfügbar sein (weil z.B. nur ein Teil einer Tabelle im Cache gehalten wird) wird das SELECT als Passthrough SQL zur Oracle Datenbank weitergeleitet. Änderungen werden immer als Passthrough SQL in der Oracle Datenbank ausgeführt. Änderungen sind sogar möglich, wenn die Oracle DB nicht online ist. Diese Änderungen werden dann später nachgeholt
In der Asynchronous Writethrough (AWT) Cache Group werden COMMITs an die Oracle Datenbank propagiert. Die Anwendung weiß also nicht, ob das COMMIT in der Oracle Datenbank erfolgreich war. Auch hier sind Änderungen möglich, wenn die Oracle DB nicht online ist. Die Änderungen werden dann später nachgeholt
In der Synchronous Writethrough (SWT) Cache Group werden COMMITs synchron an die Oracle Datenbank propagiert. Die Anwendung kann nun sicher sein, dass das COMMIT in der Oracle DB erfolgreich war, muss aber dafür auch länger warten. Die Zeitvorteile einer in-memory Verarbeitung gehen bei Änderungen verloren.
Wem diese drei Möglichkeiten nicht reichen, kann in einer User Managed Cache Group das Verhalten selbst bestimmen. So lassen sich z.B. unterschiedliche Konfigurationen pro Tabelle vergeben.

## Installation und Konfiguration

Nachdem die Architektur in den vorigen Kapiteln vorgestellt wurde, werden in diesem Kapitel die wichtigsten Punkte bei der Installation und Konfiguration einer TimesTen Datenbank erläutert.
TimesTen lässt sich unter Windows wie auch unter Linux/Unix installieren. Spezielle Hardwareanforderungen gibt es nicht. Natürlich sollte der Hauptspeicher groß genug sein.
Die Installation der Software geht recht einfach und schnell. Auch das Anlegen einer Datenbank erfordert nur wenige Schritte. Sie wird einfach als Data Source Name (DSN) angelegt. Unter Windows geht dies graphisch über den ODBC-Datenquellen-Administrator.

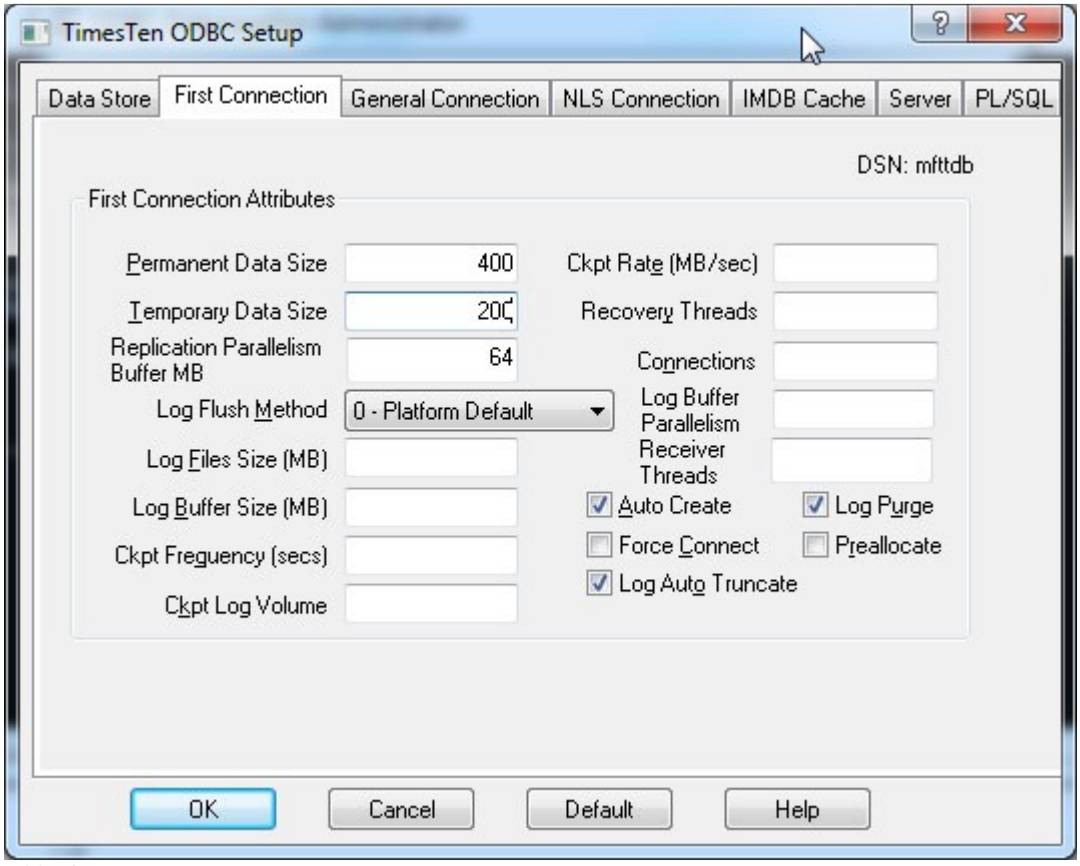

*Abb. 6: TimesTen ODBC Setup*

Wichtig sind hier die Parameter zur Einstellung der Datenbankgröße: Permanent Data Size gibt an, wie viel Speicher für die eigentlichen Daten inklusive Indexes zur Verfügung steht. Ist dieser permanente Speicher voll, lassen sich keine weiteren Daten mehr speichern. Die Temporary Data Size gibt den „Arbeitsspeicher“ an. In diesem werden Joins, Sorts usw. ausgeführt. Reicht dieser nicht aus, bricht die Verarbeitung mit einer Fehlermeldung ab.

Unter Linux/Unix sind die Werte in der Datei /home/tt/TimesTen/tt1122/info/sys.odbc.ini zu setzen:

```
[mfttdb]
# Driver = the TimesTen Direct Linked ODBC Driver
Driver=/home/tt/TimesTen/tt1122/lib/libtten.so
# DataStore = the location and the name of the database files
DataStore=/home/tt/ttdata/database/mfttdb
# LogDir = the directory for the transaction logs
LogDir=/home/tt/ttdata/logs
# PermSize = the size of the permanent region of the database in MB
PermSize=200000
# TempSize = the size of the temporary region of the database in MB
TempSize=100000
# DatabaseCharacterSet = the character set used by the database
DatabaseCharacterSet=AL32UTF8
# OracleNetServiceName = the TNS service name to Oracle db for caching
OracleNetServiceName=orcl_db
```

Abschließend ist noch der TimesTen Daemon zu starten mit `ttdaemonadmin –start` und dann kann die Datenbank verwendet werden.

**Tabellen & Indexes**

Tabellen und Indexes in TimesTen lassen sich analog zu Tabellen und Indexes in Oracle anlegen. Zusätzlich zu den Oracle Datentypen gibt es noch spezielle TimesTen Datentypen:

| Datentyp | Wertebereich | Beschreibung |
|---|---|---|
| TT_TINYINT | 0 .. 255 (2^8-1) | |
| TT_SMALLINT | -2^15 – 2^15-1 | 2-Byte Integer |
| TT_INTEGER | -2^31 – 2^31-1 | 4-Byte Integer |
| TT_BIGINT | -2^63 – 2^63-1 | 8-Byte Integer |
| TT_DATE | 1753-01-01<br>bis 9999-12-31 | Ganze Tage (ohne Uhrzeit)<br>Format YYYY-MM-DD (4 Bytes) |
| TT_TIMESTAMP | 1753-01-01 00:00:00<br>bis<br>9999-12-31 23:59:59 | Datum und Zeit mit Millisekunden<br>Weniger Speicherbedarf und schneller als TIMESTAMP, aber kleinerer Wertebereich |

*Abb. 7: TimesTen Datentypen*

Die Vorteile bei der Verwendung der TimesTen Datentypen liegen in einem geringeren Speicherverbrauch (bei allerdings eingeschränktem Wertebereich) und besserer Performance.

In TimesTen werden nämlich Zeilen immer mit einer festen Länge gespeichert, selbst wenn diese VARCHAR2 Felder enthalten. Bei den VARCHAR2 Feldern gibt es noch eine weitere Besonderheit: Inline- und Out-of-Line-Columns. Inline-Columns haben wie oben beschrieben eine feste Länge und werden als Datensatz in einer Zeile am Stück gespeichert. Out-of-Line - Columns haben eine variable Länge und werden nicht innerhalb der Zeile gespeichert. Hier sinkt der Speicherbedarf, aber dafür erhöht sich die Zugriffszeit. Per Default werden VARCHAR2, NVARCHAR2 und VARBINARY Columns mit eine Länge von mehr als 128 Byte als Out-of-Line-Column angelegt. Alle LOB-Datentypen werden immer out-of-Line gespeichert.

TimesTen bietet drei verschiedene Index-Arten an:

- Range Index
- Hash Index
- Bitmap Index

Ein Range Index kann UNIQUE sein und auch mehrere NULL-Werte enthalten. Auf ihm sind Index-Range-Scans möglich.
Der Hash Index dagegen eignet sich nicht für Index-Range-Scans, er ist dagegen sehr schnell für *exact match lookups*. Im Vergleich zu dem Range Index benötigt er in der Regel mehr Speicherplatz. Pro Tabelle kann es maximal einen Hash Index geben.
Der Bitmap Index in TimesTen funktioniert analog zu dem Bitmap Index in der Oracle Datenbank. Er ist für Spalten mit wenigen Wertausprägungen (im Vergleich zur Anzahl der Zeilen) geeignet und wird oft im Data Warehouse Bereich verwendet.

Für den Cost-based-Optimizer der TimesTen Datenbank sind Statistiken genauso wichtig wie in der Oracle Datenbank. Diese müssen explizit erstellt werden. Dazu dienen die Prozeduren `ttOptUpdateStats` und `ttOptEstimateStats`.

**PL/SQL**

TimesTen unterstützt auch PL/SQL mit Features wir Cursors und Bulk Operations. Allerdings werden stehen nicht alle Packages wie in der Oracle Datenbank zur Verfügung, sondern nur folgende Packages:

| TimesTen Packages |
|---|
| DBMS_LOB |
| DBMS_LOCK |
| DBMS_OUTPUT |
| DBMS_PREPROCESSOR |
| DBMS_RANDOM |
| DBMS_SQL |
| DBMS_STANDARD |
| DBMS_SYS_ERROR |
| DBMS_SYS_SQL |
| DBMS_UTILITY |
| UTL_FILE |
| UTL_IDENT |
| UTL_RAW |
| UTL_RECOMP |

**SQLDeveloper und TimesTen**

TimesTen bringt keine eigenen graphischen Tools mit, aber dafür unterstütz der Oracle SQLDeveloper TimesTen:

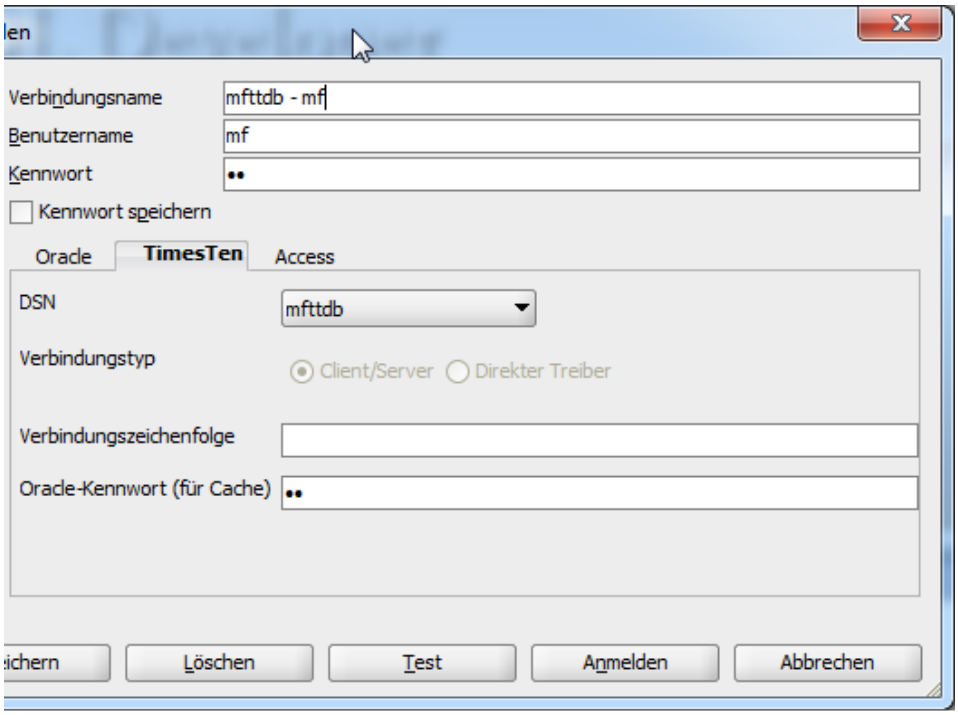


*Abb. 8: TimesTen Verbindung im SQLDeveloper*

Neben den erwarteten Dingen wie Schema-Browser und SQL-Worksheet, wird doch auch einiges an TimesTen-Spezifika unterstützt. So lässt sich zum Beispiel komfortabel die Größe von Tabellen berechnen:

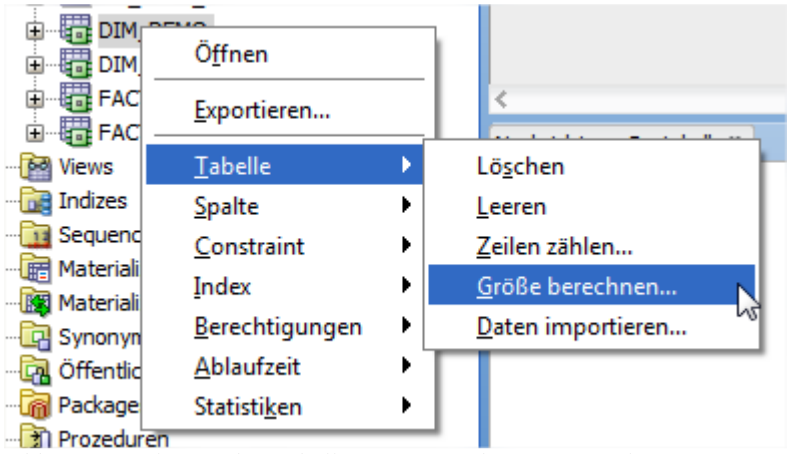


*Abb. 1: Berechnung der Tabellengröße mit dem SQLDeveloper*

Und – wie bei Oracle üblich – lässt sich auch das dafür verwendete Kommando anzeigen:

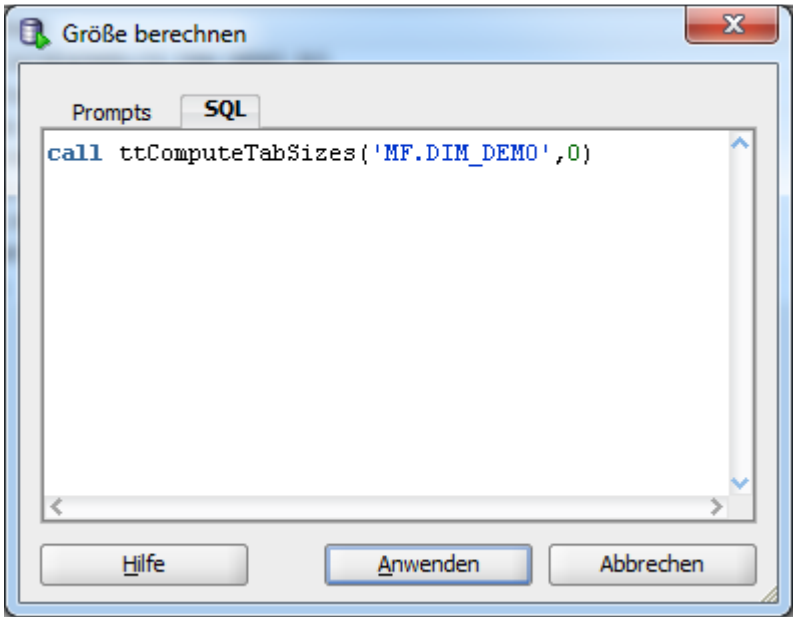


*Abb. 9: TimesTen Verbindung im SQLDeveloper*

Ferner wird auch das Arbeiten mit Cache Groups unterstützt:

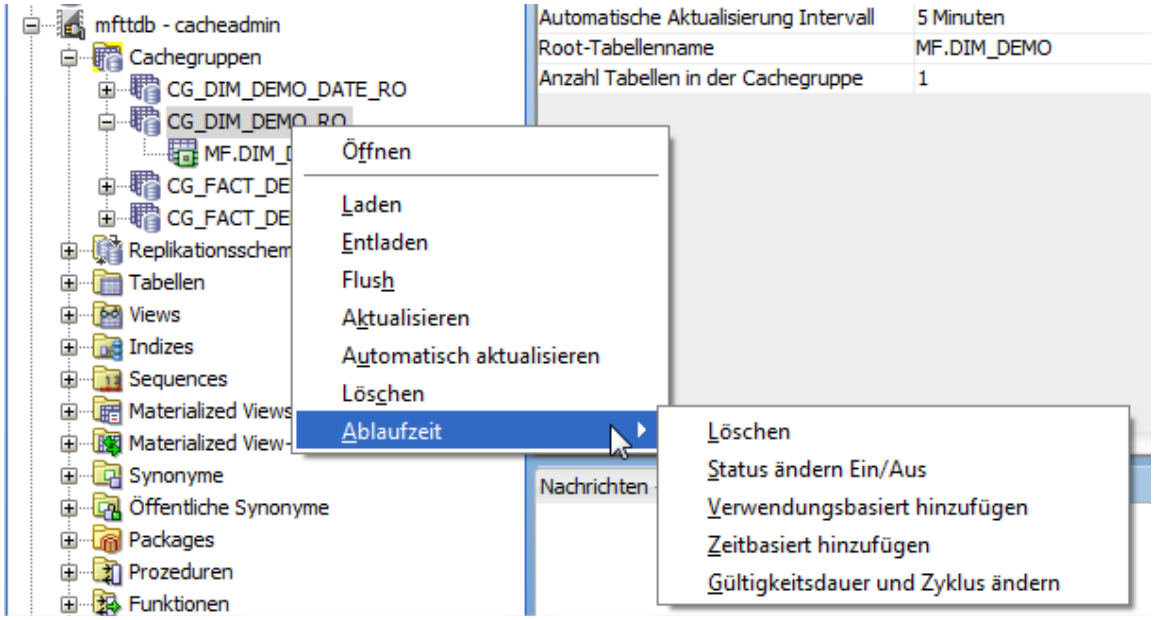


*Abb. 10: Cache Groups im SQLDeveloper*

**Vergleich Oracle, TimesTen und TimesTen for Exalytics**

Wie bereits schon erwähnt, gibt es nicht nur Unterschiede zwischen der Oracle Datenbank und TimesTen, sondern auch zwischen TimesTen und TimesTen for Exalytics. Die folgende Tabelle gibt einen Überblick über ein paar ausgewählte Features:

| Feature | Oracle | TimesTen | Exalytics |
|---|---|---|---|
| Partitioning | ✓ | ✗ | ✗ |
| Flashback | ✓ | ✗ | ✗ |
| Parallel | ✓ | ✗ | ✗ |
| Columnar Compression | ✗ | ✗ | ✓ |
| Table Compression | ✓ | ✗ | ✗ |
| Analytische Funktionen | ✓ | ✓ | ✓ |
| GROUPING SETS / ROLLUP / CUBE | ✓ | ✗ | ✓ |
| External Tables | ✓ | ✗ | ✗ |
| Materialized Views | ✓ | ✓ | ✓ |
| Bitmap Indexes | ✓ | ✓ | ✓ |
| WITH-Clause | ✓ | ✓ | ✓ |
| Skalare Subqueries | ✓ | ✗ | ✗ |
| LISTAGG | ✓ | ✗ | ✗ |
| Hierarchical Queries | ✓ | ✗ | ✗ |
| Oracle Text | ✓ | ✗ | ✗ |
| OWB Target (durch Code Templates) | ✓ | ✗ | ✗ |
| ODI Target | ✓ | ✓ | ✓ |

*Abb. 11: Feature Vergleich*

Im Unterschied zur „normalen“ TimesTen bietet *TimesTen for Exalytics* Columnar Compression (spaltenweise Komprimierung ähnlich zur Exadata) sowie die Gruppierungsfunktionen GROUPING SETS, ROLLUP und CUBE.

## Fazit

Als Datenbank für operative Anwendungen (OLTP) ist TimesTen schon lange gut gerüstet. Gerade die Möglichkeit des direkten Speicherzugriffs über Shared Libraries ist ein sehr interessantes Feature für Anwendungen mit vielen zeitkritischen und kurzen Transaktionen, bei denen sich der Wegfall des Network-Overheads durchaus bemerkbar macht.

Aber als Datamart bietet TimesTen schon recht viele interessante Features: Bitmap Indexes, analytische Funktion, etc. Leider sind manche Funktionen wie Grouping Sets, Rollup und Cube und natürlich die Columnar Compression der Exalytics vorbehalten. Die Cache Groups sind eine sehr elegante Möglichkeit, die Daten aus dem DWH in den TimesTen Datamart zu laden. Hierbei müssen dann aber gegebenenfalls die Faktentabellen um einen künstlichen Primary Key erweitert werden. Schade ist, dass TimesTen noch kein paralleles SQL unterstützt, da dieses sicherlich noch Potential für die Auswertung großer Datenmengen birgt.

Alles in allem sollte man den Einsatz von TimesTen aber durchaus in Betracht ziehen, besonders bei Neuentwicklungen. Bestehende Systeme nach TimesTen zu migrieren ist sicherlich möglich, aber auch mit einem gewissen manuellen Aufwand verbunden.

## Kontaktadresse:

Carsten Herbe
Metafinanz-Informationssysteme GmbH
Leopoldstr. 146
D-80804 München

Telefon: +49 (0) 360531 5039
Fax: +49 (0) 360531 15
E-Mail carsten.herbe@metafinanz,de
Internet: www.metafinanz.de