Oracle® Database

Getting Started with Cache Guide

Release 26.1
F92275-01
March 2026

Oracle Database Getting Started with Cache Guide, Release 26.1

F92275-01

Primary Author: Sheryl Maring

Contents

7 Create Read-Only Cache Groups

8 Create Static Asynchronous WriteThrough Cache Groups

9 Next Steps

1

Paths to Explore Cache in TimesTen

TimesTen cache is a robust feature with many concepts and options for caching data between a TimesTen database and an Oracle database. When you are starting to learn about caching within TimesTen, there are three learning paths:

- Accelerate your Applications - Achieve Blazing Fast SQL With an Oracle TimesTen Cache. LiveLab: This LiveLab will help you to become familiar with setting up and using a TimesTen cache. The lab focuses briefly on concepts and heavily on the tasks for how to quickly configure, create, and use a TimesTen cache.
- *Oracle TimesTen In-Memory Database Getting Started with Cache Guide*: After going through the LiveLab, use this getting started guide for an overview of the basics of cache provided by TimesTen. This guide covers the most popular options and uses the default options when showing you how to create the three most popular cache group types. Start with Overview of Cache.
- *Oracle TimesTen In-Memory Database Cache Guide*: After you are familiar with the basics of caching, you can see a full explanation of concepts and details of the TimesTen cache feature. This is the advanced guide for cache. Start with Overview of Cache.

2

Overview of Cache

Caching in TimesTen provides the ability to transfer data between an Oracle database and a TimesTen database. You can cache Oracle Database data and reduce the workload on the Oracle database. You can configure for read-only or read-write caching for Oracle database tables.

Use caching to improve the performance for your applications access to data. Caching offloads computing cycles from Oracle databases and enables responsive and scalable applications.

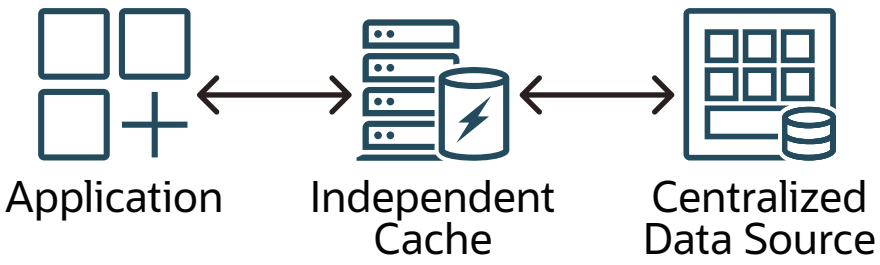
A cache in TimesTen:

- Loads a subset of the Oracle database tables into a TimesTen database.
- Can automatically maintain data synchronization between the cached data in the TimesTen database and the source data in the Oracle database.

The following sections describe the basics of cache in TimesTen:

What is Caching?

In an ideal world, direct access to authoritative data would always be fast and scalable. Most applications access data as a client where the centralized data source is in another location, which can lead to decreased performance, scalability, and availability. Alternatively, you could place an independent cache in an optimal location with a subset of relevant data cached from the centralized data source to increase your performance, scalability, and availability.
An independent cache can be added in front of a centralized data source (in our case, an Oracle database).



- You can store copies of data derived from a centralized data source into one or more independent caches, which can be located *closer* to where data consumption occurs. This improves data access latency when used on read-only data that is accessed frequently. Lower latency means that there is a minimal delay in the transporting of data over a network connection resulting in smaller overall delay times.

- An independent cache uses high-speed RAM for temporarily storing subsets of data. Future requests for that same data are served up faster than is possible by accessing the data from a centralized data source. The data on a centralized data source is typically stored on disk, which is also impacted by high rates of access.
- You can store cached data for long periods of time or just temporarily.
- You can keep the amount of data on the independent cache as small as possible while the data is still relevant to what is needed.

Since the data stored in the independent cache are copies of the data on the centralized data source, there are mechanisms to ensure that the cached data remains synchronized with the centralized data source.

There are two ways to use an independent cache:

- **As a read-only cache:** Data is modified only on the centralized data source and these changes are propagated from the Oracle database to the independent cache as specified.
- **As a read-write cache:** Data can be modified on either the independent cache or on the centralized data source. The data is propagated to either the independent cache or to the centralized data source as specified.

The specific actions for read-only and read-write operations within the independent cache are:

Actions for caching	The specific actions for read-only operations within the independent cache	The specific actions for read-write operations within the independent cache
What can be changed?	Data can be changed at any time on the centralized data source. Data is not changed on the independent cache.	Data can be changed at any time on the independent cache. While it is possible, data should not be changed on the centralized data source.
How is the data in the independent cache and the centralized data source synchronized?	Any changes made to the data on the centralized data source are propagated up from the centralized data source to the independent cache.	Any changes made to the data on the independent cache can be manually or dynamically flushed down to the centralized data source.

You use SQL statements to define what data is to be cached and how changes are propagated.

- Read-Only operations use the `SELECT` SQL statement. For example:

```
SELECT * FROM sales.customers;
< 342, West, Jane Stone >
< 663, East, Pat Reed >
2 rows found.
```

- Read-Write operations use `SELECT`, `INSERT`, `UPDATE`, and/or `DELETE` SQL statements. For example:

```
INSERT INTO customers VALUES (342, "West", "Jane Stone");
1 row created.
DELETE FROM customers WHERE cust_num=122;
I row deleted.
UPDATE customers SET region="East" WHERE cust_num=663;
1 row updated.
```

Using TimesTen as a Cache

The TimesTen database acts as an independent cache for an Oracle database (that acts as the centralized data source). The TimesTen database caches data from an Oracle database. The TimesTen database provides support for read-only or read-write transactions in its operation as an independent cache.

Learn About Cache Objects

A TimesTen database acts as an independent cache to cache data from an Oracle database. A **cache group** is a TimesTen database object that enables caching of Oracle database tables.

About Cache Groups and Cache Tables

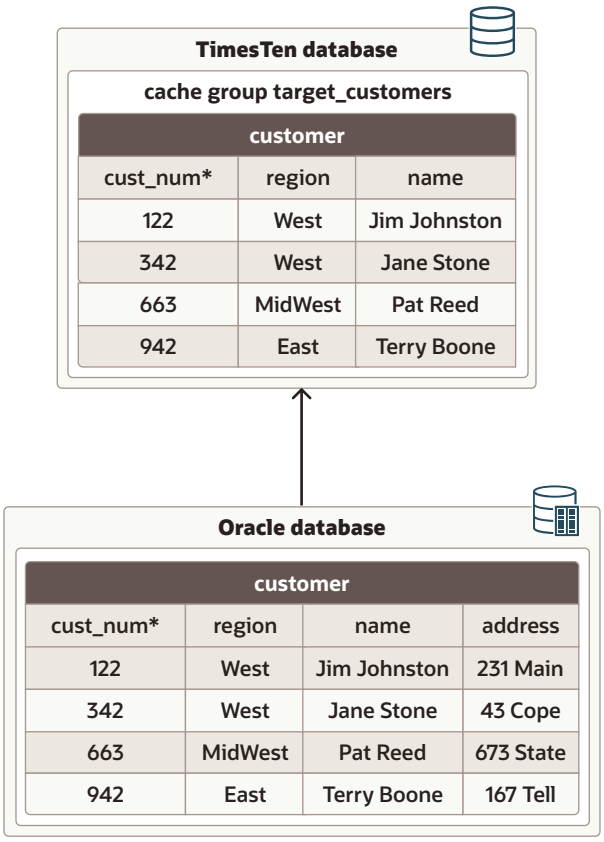
You use cache groups to define which data is to be cached in a TimesTen database from an Oracle database. A cache group can be defined to cache the whole of or just a part of a single Oracle database table or a set of related Oracle database tables. A single TimesTen database can contain multiple cache groups.
A cache group is a SQL object that encapsulates a set of one or more tables that are related through primary key and foreign key relationships. The single top-level table is called the root table and the other tables are below it in a hierarchical parent/child arrangement.

When you create a cache group, you specify one cache table for each Oracle table represented in the cache group. Each Oracle table must have a unique key (either a primary key or a unique index defined over one or more `NOT NULL` columns). However, you must include all columns of the unique key. You can define all or a subset of the columns of the Oracle table within its cache table when creating a cache group (subsets specified with a `WHERE` clause).

Decide on the following when creating a cache group that specifies one or more cache tables on the Oracle database.

1. Identify the schema or schemas and tables on the Oracle database that you want to cache in the TimesTen database. You are not required to cache all of the tables in each schema.
2. Create a cache group on the TimesTen database that specifies the Oracle tables that you want cached in the TimesTen database.

The figure below shows a fictional cache group called `customer_orders` that was created on the TimesTen database to cache a subset of the data that exists within the `customers` table located on the Oracle database. The `customers` table is a single table with no children. A subset of data in the `customers` table on the Oracle database is cached in the `customer_orders` cache group on the TimesTen database.



When caching Oracle database tables:

- You can have a cache group that caches only a single Oracle database table. In a single-table cache group, there is a root table but no child tables.
- You can cache multiple related Oracle database tables in the same cache group by defining a root table and one or more child tables. A cache group can contain only one root table. The root table does not reference any table with a foreign key constraint.

In a cache group with multiple tables, each child table must reference the root table or another child table in the same cache group using a foreign key constraint. Cache tables defined in a multiple-table cache group must be related to each other in a TimesTen database through foreign key constraints. However, the corresponding tables in the Oracle database do not necessarily need to be related to each other.

The tables on the Oracle database can be related:

- Through an explicit foreign key constraint.
- Without an explicit foreign key constraint. You may have tables on the Oracle database that are not related through a foreign key constraint. However, you want to cache the data within these separate tables on the TimesTen database. The user application could maintain a relationship between tables that is not enforced by foreign key constraints on the Oracle database.

About Cache Instances

A cache instance is defined as a single row in the cache group's root table together with the set of related rows in the child tables.

A cache instance is the smallest unit of data transferred when data is loaded from an Oracle database into a cache group in a TimesTen database.

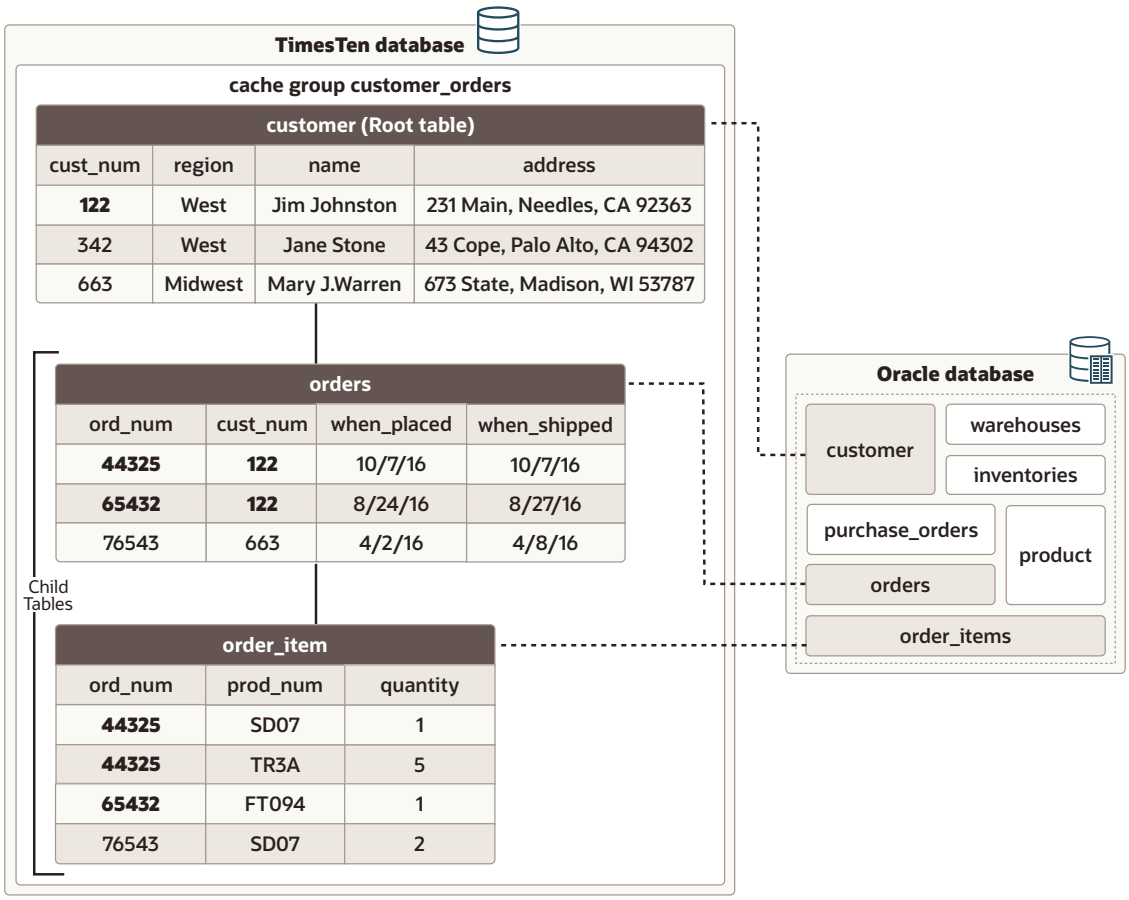
Cache operations act on cache groups (not on individual tables) and on cache instances (not individual rows). Normal SQL operations, such as `SELECT`, `INSERT`, `UPDATE` and `DELETE`, operate directly on the cache tables and their rows.

The following graphic shows three cache tables in the `customer_orders` cache group. This cache group caches data from the `customers`, `orders`, and `order_items` tables in the Oracle database. It shows multiple related Oracle database tables in the same cache group with a root table and one or more child tables. Cache tables defined in a multiple-table cache group must be related to each other in a TimesTen database through foreign key constraints.

- The root cache table is `customers`.
- The `orders` and `order_item` cache tables are child tables. The `orders` cache table has a foreign key of `cust_num` to show the relationship with the `customer` cache table. The `order_item` cache table has a foreign key of `ord_num` to show the relationship with the `orders` cache table.

Because of the relationships between the cache tables, the cache instance identified by the row with the value 122 in the `cust_num` primary key column of the `customers` table includes:

- The two rows with the value 122 in the `cust_num` column of the `orders` cache table (whose value in the `ord_num` primary key column is 44325 or 65432), and
- The three rows with the value 44325 or 65432 in the `ord_num` column of the `order_item` cache table

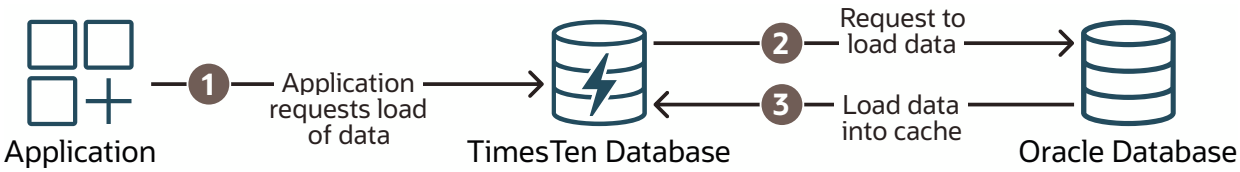


Synchronizing Data Between TimesTen and Oracle Databases

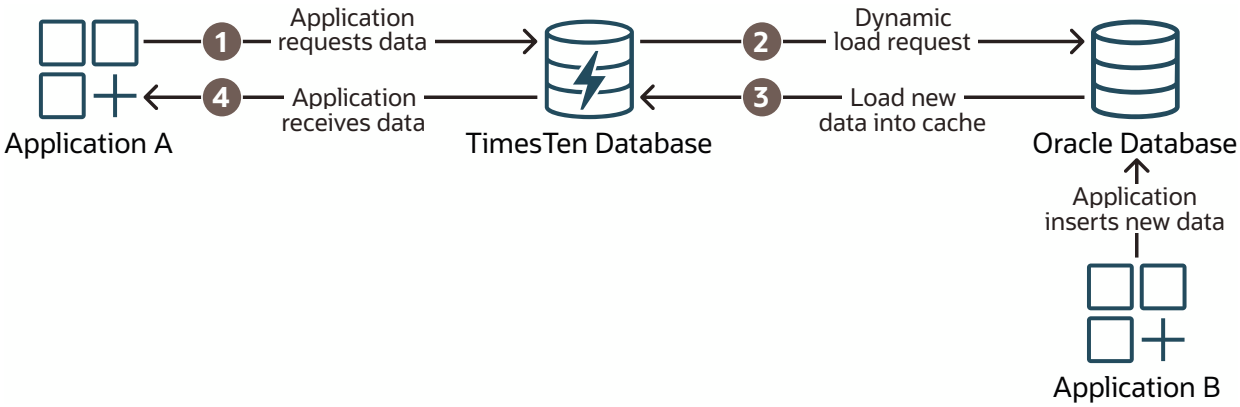
Data is sychronized between the TimesTen database and the Oracle database through load, refresh, or propagate operations. The load and refresh operations can be performed manually or configured to occur automatically.

- Load operations:
  - Static load: A static load operation manually loads new cache instances into cache tables on a TimesTen database from the Oracle database tables using a `LOAD CACHE GROUP` SQL statement. The `LOAD CACHE GROUP` SQL statement loads committed inserts on the Oracle database tables into the cache tables on a TimesTen database.

    A load operation is primarily used to initially populate a cache group. When you first set up a cache within a TimesTen database, some cache group types require performing a manual load of the subset of data that you want to be available in the cache. This creates a copy of the data that exists on the Oracle database. This is true for both read-only and read-write cache groups except for cache groups that use dynamic load. (You will learn about dynamic load later.)

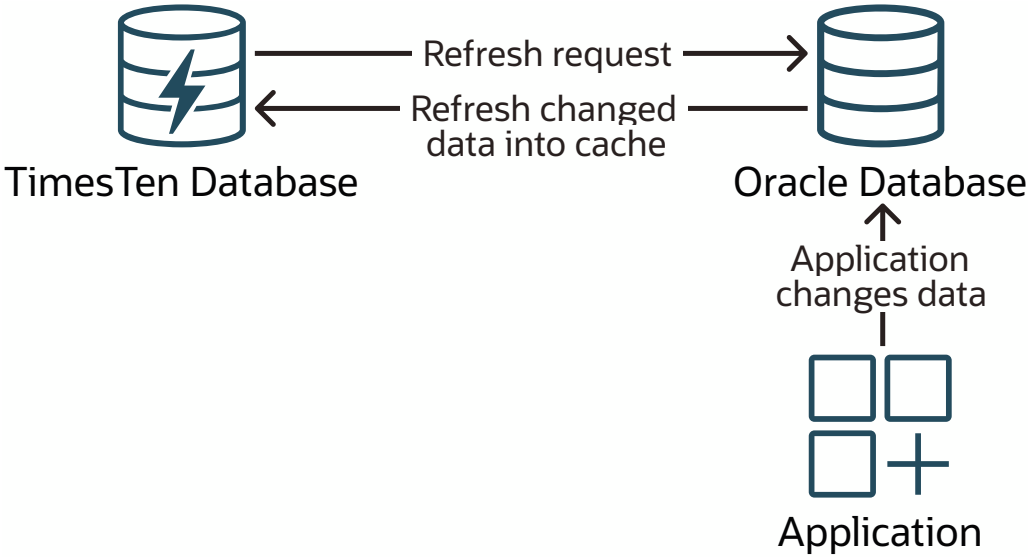


  - Dynamic load: When the application requests data that does not exist in the TimesTen database, the TimesTen database has mechanisms to dynamically retrieve and load the new data from the Oracle database into the TimesTen database. The application dynamically requests data with a `SELECT` SQL statement with a `WHERE` clause specifying the data.

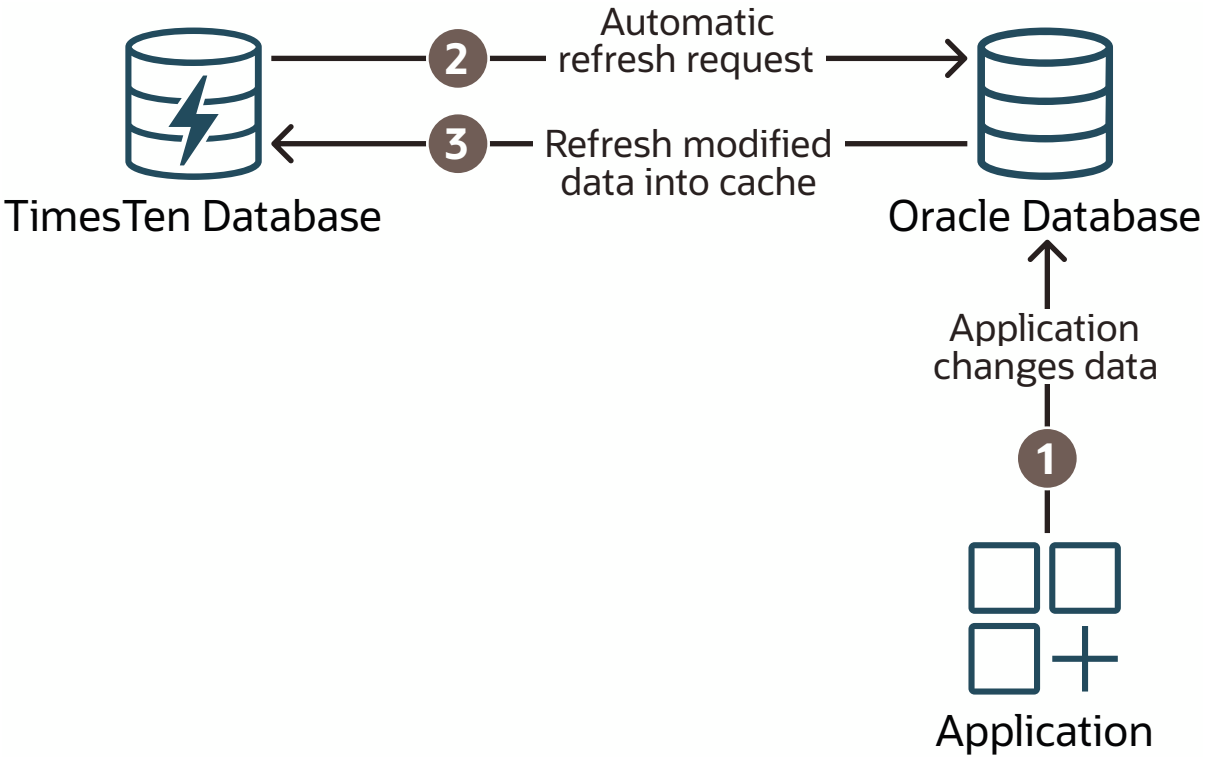


- Refresh operations: These operations are only for read-only caching, data can be modified at any time on the Oracle database with `INSERT`, `UPDATE` or `DELETE` SQL statements. A refresh operation synchronizes changed data from an Oracle database to cache table in a TimesTen database. Thus, cache instances in the cache tables on a TimesTen database contain the latest copy of the data from the Oracle database tables. A refresh operation is primarily used to apply committed changes on the Oracle database tables to existing cache tables after the cache group has been initially populated. You can request a refresh manually or automatically at a specified time interval.

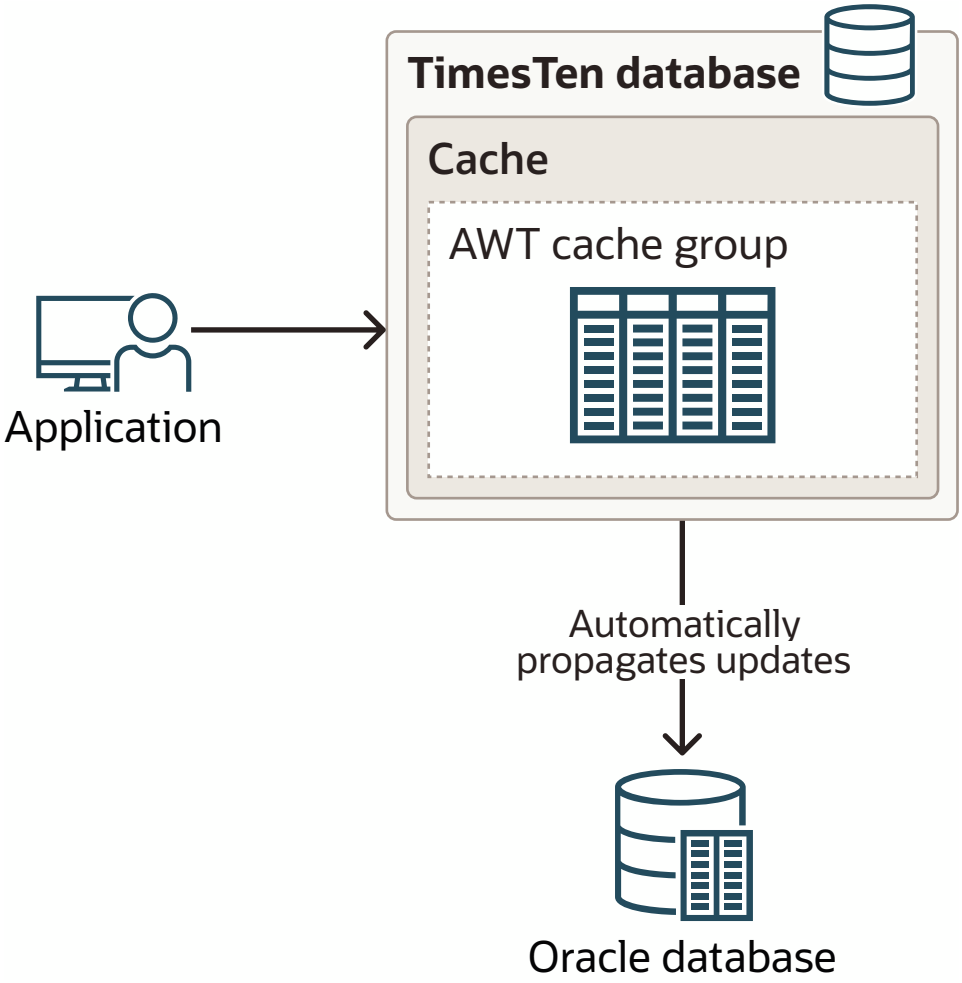
– Manual refresh: For a manual refresh operation, run the `REFRESH CACHE GROUP` SQL statement.



– Automatic refresh (autorefresh) operations: TimesTen provides mechanisms so that you can set an interval of time after which TimesTen automatically refreshes (known as autorefresh) all changes from the Oracle database to the TimesTen database. If you have specified autorefresh for your read-only caching, then the cached data is automatically synchronized at a specified interval of time. Normally, if you have autorefresh specified, you will not use manual refresh unless you want to refresh in between the autorefresh interval.



- Automatic propagation of changes from the TimesTen database to the Oracle database: This operation is only available for read-write caching. When you change data on the cache tables on TimesTen, committed changes are automatically propagated to the cached Oracle Database tables.



Types of Cache Groups You Can Create

TimesTen provides multiple types of cache groups that cache Oracle database tables with automatic data synchronization. Select the cache group type based on your application needs. This guide covers the three most popular cache group types.

The cache group type identifies if you want to only read from or be able to read and write to the cache tables.

- **Read-Only cache group**: A read-only cache group enforces a caching behavior in which committed changes on cached tables in the Oracle database are automatically refreshed to the cache tables in the TimesTen database. Using a read-only cache group is suitable for reference data that is heavily accessed by applications.
  - Static read-only cache group: With a static read-only cache group, you use manual load requests to load data. You can use manual refresh requests. However, most read-only cache groups use autorefresh operations to refresh modified data at specified

time intervals. When using autorefresh, the manual refresh requests are not necessary.

  - Dynamic read-only cache group: With a dynamic read-only cache group, the application relies on data dynamically loading when data is requested with a qualified `SELECT... WHERE` SQL statement. Most read-only cache groups use autorefresh operations to refresh modified data at specified time intervals.

- **Read-write cache group**: This guide describes the most popular read-write cache group that TimesTen offers with the static **Asynchronous WriteThrough (AWT)** cache group. A static AWT cache group enforces a caching behavior in which committed changes on cache tables in the TimesTen database are automatically propagated and committed to the cached tables in the Oracle database in asynchronous fashion. Using an AWT cache group is suitable for high speed data capture and online transaction processing.

3
Before You Begin

This quick start cache guide covers setting up a cache environment after you have already created a TimesTen installation, created a TimesTen instance, and started the main daemon.

This guide assumes that you are already familiar with some of the basic TimesTen concepts. In addition, since cache includes connecting to the Oracle database, you need to know how to configure for connecting to an Oracle database.

TimesTen Concepts	**More Information**
TimesTen instance and instance administrator	Overview of Installations and Instances in the *Oracle TimesTen In-Memory Database Installation, Migration, and Upgrade Guide*
Data Source Name (DSN)	Specifying Data Source Names to Identify TimesTen Databases in the *Oracle TimesTen In-Memory Database Operations Guide*
Connection attributes	Connection Attributes in the *Oracle TimesTen In-Memory Database Reference*
Permanent and temporary memory configured with the `PermSize` and `TempSize` connection attributes	Specifying the Memory Region Sizes of a Database in the *Oracle TimesTen In-Memory Database Operations Guide*
The ttIsql utility used for connecting to the TimesTen database	ttIsql in the *Oracle TimesTen In-Memory Database Reference*

In addition, you need to know how to set up an Oracle Net service name in a `tnsnames.ora` file that is used to connect to your Oracle database.

Connecting to an Oracle Database	**More Information**
To learn more about configuring the `tnsnames.ora` file and Oracle Net service name	Configuring Naming Methods in *Oracle Database Net Services Administrator's Guide*

4

Prepare the Databases for Cache Operations

Before you can start cache operations, you must prepare both the Oracle database and TimesTen database for cache operations.

The following sections describe what you need to do in order to prepare both the Oracle and TimesTen databases for cache operations.

- Prepare the Oracle Database for Cache Operations
- Prepare the TimesTen Database for Cache Operations

Prepare the Oracle Database for Cache Operations

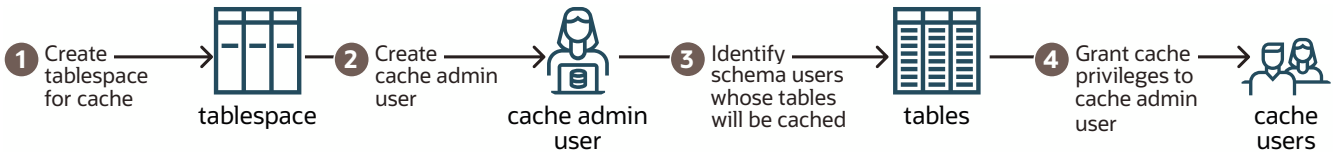
As a prerequisite for setting up the TimesTen database as a cache for tables from an Oracle database, you need to configure the Oracle database for cache operations. In order to configure the Oracle database for cache operations, you need database administrator access.

Connect to the Oracle database as a database administrator to configure the Oracle database for cache operations. This example uses the SQL*Plus utility:

```
% sqlplus sys@tnsservicename as sysdba
Enter password: password
```

> **Note**
>
> The use of the `sys@tnsservicename as sysdba` user in this example is applicable only for a test environment.

The following shows the steps necessary for preparing the Oracle database for cache operation:



> **Note**
>
> If you are using Oracle Autonomous Database Serverless or Oracle Autonomous Database on Dedicated Exadata Infrastructure for the Transaction Processing workload type, use the preconfigured databases services LOW or TP. In addition, if you are using a multitenant container database (CDB) or pluggable database (PDB), note the specific instructions below on how to create the cache administration user and grant this user privileges in a CDB or PDB.

Task 1: Create the Tablespace for Cache Metadata and Management Objects

The Oracle cache administration user needs to have a default tablespace to store cache metadata and management objects. It is recommended to create a dedicated tablespace just for cache operations. This tablespace should not be shared with other applications.

The following example creates the `cachetblsp` tablespace in the non-autonomous Oracle Database:

```
SQL> CREATE TABLESPACE cachetblsp DATAFILE 'cachetblsp_f1.dbf'
                SIZE 5G SEGMENT SPACE MANAGEMENT AUTO;
Tablespace created.
```

Skip this step for the Autonomous Transaction Processing. Autonomous Transaction Processing automatically configures default data and temporary tablespaces for the database. Adding, removing, or modifying tablespaces is not allowed. Autonomous Transaction Processing creates one or multiple tablespaces automatically depending on the storage size.

Task 2: Create the Cache Administration User on the Oracle Database

To be able to cache tables from an Oracle database to a TimesTen database, create a **cache administration user on the Oracle database** on both the Oracle and TimesTen databases.

The cache administration user on the Oracle database creates and maintains Oracle database objects that store the information used to manage the cache environment.

The following example creates the `cacheadmin` user (with password `cacheadmpwd`) in the non-autonomous Oracle Database and specifies the default tablespace for cache management objects, `cachetblsp`:

```
SQL> CREATE USER cacheadmin IDENTIFIED BY cacheadmpwd
                DEFAULT TABLESPACE cachetblsp QUOTA UNLIMITED ON cachetblsp;
User created.
```

For the Transaction Processing workload type, the following example creates the cache administration user:

```
SQL> CREATE USER cacheadmin IDENTIFYED BY cacheadmpwd QUOTA UNLIMITED ON DATA;
```

Later on, you will create a cache administration user on the TimesTen database with the same name to coordinate the caching of data between the Oracle and TimesTen databases.

Task 3: Identify Schema Users on the Oracle Database

To be able to cache tables from an Oracle database to a TimesTen database, you need to identify (or create) one or more **schema users** on the Oracle database.

These schema users own tables that will be cached in the TimesTen database.

> **Note**
>
> The set of examples included in this guide assume that a schema user (with populated tables) named `sales` already exists in the Oracle database.

Later on, you will create one or more cache table users on the TimesTen database with the same name as the schema users to coordinate the caching of data between the Oracle and TimesTen databases.

Task 4: Grant SQL Privileges to the Cache Administration User on the Oracle Database

The cache administration user on the Oracle database requires a specific set of SQL privileges to perform cache operations in the Oracle database.

Run the `grantCacheAdminPrivileges.sql` script as the Oracle database administrator to grant to the Oracle cache administration user the minimum set of SQL privileges required to perform cache operations. The script is available at *`timesten_home`*`/install/oraclescripts/grantCacheAdminPrivileges.sql` of your TimesTen instance.

The following example for a non-autonomous Oracle Database grants the required SQL privileges to the `cacheadmin` user for cache operations in the Oracle database:

```
SQL> @grantCacheAdminPrivileges.sql cacheadmin


Please enter the administrator user id
The value chosen for administrator user id is cacheadmin

***************** Creation of TT_CACHE_ADMIN_ROLE starts ******************
0. Creating TT_CACHE_ADMIN_ROLE role
** Creation of TT_CACHE_ADMIN_ROLE done successfully **
***************** Initialization for cache admin begins ******************
0. Granting the CREATE SESSION privilege to CACHEADMIN
1. Granting the TT_CACHE_ADMIN_ROLE to CACHEADMIN
2. Granting the DBMS_LOCK package privilege to CACHEADMIN
3. Granting the DBMS_DDL package privilege to CACHEADMIN
4. Granting the DBMS_FLASHBACK package privilege to CACHEADMIN
5. Granting the CREATE SEQUENCE privilege to CACHEADMIN
6. Granting the CREATE CLUSTER privilege to CACHEADMIN
7. Granting the CREATE OPERATOR privilege to CACHEADMIN
8. Granting the CREATE INDEXTYPE privilege to CACHEADMIN
9. Granting the CREATE TABLE privilege to CACHEADMIN
10. Granting the CREATE PROCEDURE  privilege to CACHEADMIN
11. Granting the CREATE ANY TRIGGER  privilege to CACHEADMIN
```

```
12. Granting the GRANT UNLIMITED TABLESPACE privilege to CACHEADMIN
13. Granting the DBMS_LOB package privilege to CACHEADMIN
14. Granting the SELECT on SYS.ALL_OBJECTS privilege to CACHEADMIN
15. Granting the SELECT on SYS.ALL_SYNONYMS privilege to CACHEADMIN
16. Checking if the cache administrator user has permissions on the default
tablespace
     Permission exists
18. Granting the CREATE TYPE privilege to CACHEADMIN
19. Granting the SELECT on SYS.GV$LOCK privilege to CACHEADMIN
20. Granting the SELECT on SYS.GV$SESSION privilege  to CACHEADMIN
21. Granting the SELECT on SYS.DBA_DATA_FILES privilege  to CACHEADMIN
22. Granting the SELECT on SYS.USER_USERS privilege  to CACHEADMIN
23. Granting the SELECT on SYS.USER_FREE_SPACE privilege  to CACHEADMIN
24. Granting the SELECT on SYS.USER_TS_QUOTAS privilege  to CACHEADMIN
25. Granting the SELECT on SYS.USER_SYS_PRIVS privilege  to CACHEADMIN
26. Granting the SELECT on SYS.V$DATABASE privilege  to CACHEADMIN (optional)
27. Granting the SELECT on SYS.GV$PROCESS privilege  to CACHEADMIN (optional)
28. Granting the SELECT ANY TRANSACTION privilege to CACHEADMIN
29. Creating the TTCACHEADM.TT_07_ARDL_CG_COUNTER table
30. Granting SELECT privilege on TTCACHEADM.TT_07_ARDL_CG_COUNTER table to
PUBLIC
********* Initialization for cache admin user done successfully *********
```

For Autonomous Transaction Processing, Step 16 output is as follows:

```
16. Checking if the cache administrator user has permissions on the default
tablespace

No existing permission.
```

Autonomous Transaction Processing automatically configures tablespaces. Therefore, this permission is not necessary.

The cache administration user on the Oracle database also needs specific privileges on each user table that is cached in TimesTen. The exact privileges depend on the type of cache groups being used, such as:

- For read-only cache groups, grant the `SELECT` privilege on all the user tables that will be cached to the cache administration user.

  ```
  SQL> GRANT SELECT ON sales.customers TO cacheadmin;
  ```

- For read-write (AWT) cache groups, the cache administration user needs `SELECT`, `INSERT`, `UPDATE`, and `DELETE` privileges on all the user tables that will be cached,

  ```
  SQL> GRANT SELECT, INSERT, UPDATE, DELETE ON sales.customers TO cacheadmin;
  ```

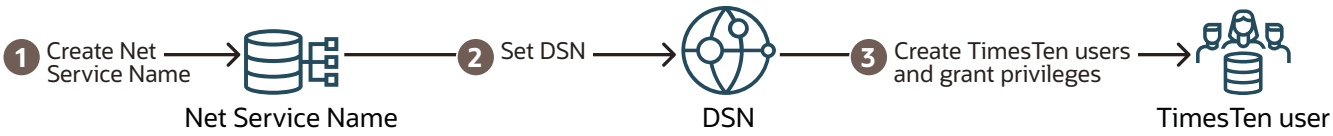
Later on when you are creating a DSN within TimesTen, you will need to know the Oracle database character set, which must match on both the Oracle and TimesTen databases.

```
SQL> SELECT value FROM nls_database_parameters WHERE parameter='NLS_CHARACTERSET';
VALUE
--------------------------------------------------------------------------------
AL32UTF8
```

Prepare the TimesTen Database for Cache Operations

Likewise, you need to prepare the TimesTen database for cache operations.

You need to perform the following actions for a TimesTen database to be able to cache data from an Oracle database.



These steps are described in the following sections:

- Task 1: Set the Net Service Name for the Oracle Database in the tnsnames.ora File
- Task 2: Create a Data Source Name (DSN) for the TimesTen Database
- Task 3: Create Users and Grant Privileges in the TimesTen Database

Task 1: Set the Net Service Name for the Oracle Database in the tnsnames.ora File

In order to connect to the Oracle database, an Oracle net service name needs to be added to the `tnsnames.ora` file.

For Autonomous Transaction Processing, use the preconfigured databases services LOW or TP:

- *`databasename`*`_tp`
- *`databasename`*`_low`

You can either create a new *`timesten_home`*`/conf/tnsnames.ora` file or copy the sample `tnsnames.ora` file from the *`timesten_home`*`/install/network/admin/samples/` directory to the *`timesten_home`*`/conf` directory.

1. Ensure that the main daemon is stopped before you modify the `tnsnames.ora` file.

   ```
   ttDaemonAdmin -stop
   ```

2. Set the `TNS_ADMIN` location for the cache agent with the `ttInstanceModify -tnsadmin` option to set the path to the `tnsnames.ora` file. In our example, the *`timesten_home`* variable points to `/TimesTen` directory. Specify the full path to the directory where the file is located, which in this case is `/TimesTen/conf`.

   ```
   ttInstanceModify -tnsadmin /TimesTen/conf
   ```

3. For cache in TimesTen, set the `TNS_ADMIN` environment variable to indicate the full path to the directory where the `tnsnames.ora` file is located. Set this variable in the user's profile script so that it persists.

   ```
   export TNS_ADMIN=/TimesTen/conf
   ```

4. Restart the main daemon to capture this setting.

   ```
   ttDaemonAdmin -start
   ```

5. Within an editor, add the net service name for the non-autonomous Oracle Database to the `tnsnames.ora` file. The following is an example of defining `orcl` in a `tnsnames.ora` file. Save this name as you will set this net service name in the DSN.

   ```
   orcl =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = myhost)
   ```

```
    (PORT = 1521))
  (CONNECT_DATA =
    (SERVICE_NAME = myhost.example.com)))
```

For Autonomous Transaction Processing, the following is an example of defining `orcl_low` in a `tnsnames.ora` file:

```
orcl_low =
 (DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = adb.us-phoenix-1.oraclecloud.com)
     (PORT = 1521))
   (CONNECT_DATA =
     (SERVICE_NAME = orcl_low.adb.oraclecloud.com)))
```

> **Note**
>
> TimesTen supports both TCP and mTLS-based connections for Oracle Autonomous Database on Dedicated Exadata Infrastructure and only mTLS-based connections for Oracle Autonomous Database Serverless.

Task 2: Create a Data Source Name (DSN) for the TimesTen Database

A TimesTen database that caches data from an Oracle database can be referenced by either a system DSN or a user DSN.

The operating system user that installed and created the TimesTen instance is called the instance administrator. When this instance administrator connects using a DSN for the first time, a TimesTen database is implicitly created.

This example is going to reference the TimesTen database with a system DSN. On UNIX or Linux, the system DSN is located in the *`timesten_home`*`/conf/sys.odbc.ini` file. As described in Connecting to a TimesTen Database, the `sys.odbc.ini` file contains the DSN definitions.

This example defines two ODBC Data Source Names (DSNs).

> **Note**
>
> ODBC is TimesTen's native API, though TimesTen also provides, or supports, many other commonly used database APIs such as JDBC, Oracle Call Interface, ODP.NET, cx_Oracle (for Python) and node-oracledb (for Node.js). See Connecting to TimesTen with ODBC and JDBC Drivers in the *Oracle TimesTen In-Memory Database Operations Guide*.

- Direct connection: The `cache1` DSN is a direct mode, or server DSN. It uses the `TimesTen 22.1 Driver`. It defines the parameters and connectivity for a database hosted by this TimesTen instance. Tools, utilities, and applications running on this host (`myhost`) can connect through this DSN using TimesTen's low latency direct mode connectivity mechanism.
- Client-server connection: This database is also accessible remotely using TimesTen's client-server connectivity. The `cache1cs` DSN is a client DSN and uses the `TimesTen 22.1 Client Driver`. It defines connectivity parameters for a server DSN that tools, utilities, and applications can connect to using TimesTen's client-server connectivity mechanism. In this

example, the client DSN, `cache1cs`, defines client-server access for the local `cache1` server DSN.

When creating a DSN for a TimesTen database that caches data from an Oracle database, the following connection attributes are important for your cache environment:

- `DataStore` specifies the fully qualified directory path name of the database and the file name prefix. This name is not a file name. In this example, `DataStore` is set to `/disk1/databases/database1`.
- `PermSize` specifies the allocated size of the database's permanent region in MB. The `PermSize` value must be smaller than the physical RAM on the machine. Set this to a value that enables you to store all of your data. The `PermSize` value could be from a few GB to several TB. This example sets the permanent region to 1024 MB.
- `TempSize` indicates the total amount of memory in MB allocated to the temporary region for the database. This example sets the temporary region to 256 MB.
- `LogBufMB` specifies the size of the internal transaction log buffer for the database. This example sets the transaction log buffer to 256 MB.
- `LogFileSize` specifies the maximum size of transaction log files in megabytes. This example sets the maximum size of transaction log files to 256 MB.
- `DatabaseCharacterSet` must match the Oracle database character set. In this example, the database character set is `AL32UTF8`.

> **Note**
>
> You can determine the Oracle database character set by running the following query in SQL*Plus as any user:
>
> ```
> SQL> SELECT value FROM nls_database_parameters
>        WHERE parameter='NLS_CHARACTERSET';
> VALUE
> --------------------------------------------------------------------------------
> AL32UTF8
> ```

- `ConnectionCharacterSet` specifies the character encoding for the connection. Generally, you should choose a connection character set that matches your terminal settings or data source. In this example, the connection character set is `AL32UTF8`.
- `OracleNetServiceName` must be set to the net service name of the Oracle database instance. This example sets this to `orcl`. This is the same name as was set in the `tnsnames.ora` file in step 1.
- `CacheAdminWallet=1` specifies that credentials for the Oracle cache administration user that are registered with the `ttCacheUidPwdSet` built-in procedure are stored in an Oracle Wallet, rather than in memory.

Then, there is an entry for the client DSN. The client DSN specifies the location of the TimesTen database with the following attributes:

- The `TTC_Server_DSN` attribute specifies the server DSN of the intended database.
- The `TTC_Server` attribute specifies the server (and the port number if you do not want to use the default port number) for the database.

In the `sys.odbc.ini` file, create a TimesTen DSN `cache1` and set the following connection attributes:

```
[ODBC Data Sources]
cache1=TimesTen 22.1 Driver
cache1cs=TimesTen 22.1 Client Driver

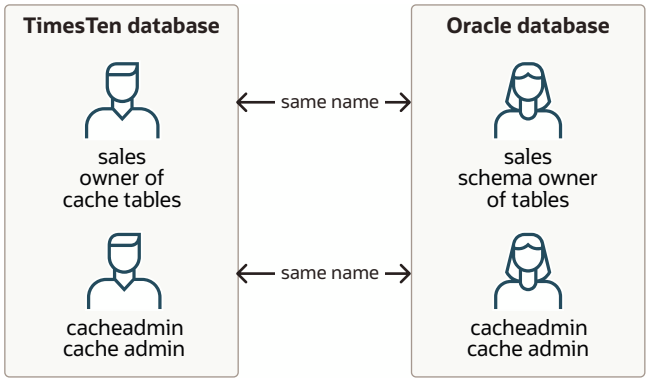
[cache1]
DataStore=/disk1/databases/database1
PermSize=1024
TempSize=256
LogBufMB=256
LogFileSize=256
DatabaseCharacterSet=AL32UTF8
ConnectionCharacterSet=AL32UTF8
OracleNetServiceName=orcl
CacheAdminWallet=1

[cache1cs]
TTC_SERVER_DSN=CACHE1
TTC_SERVER=myhost/6625
ConnectionCharacterSet=AL32UTF8
```

You can use the default settings for all the other connection attributes.

Task 3: Create Users and Grant Privileges in the TimesTen Database

In addition to the Oracle database users, you must create the following TimesTen users before you can use cache.



- A TimesTen *cache administration user* performs cache group operations. The TimesTen cache administration user must have the same name as the Oracle cache administration user that can access the cached Oracle Database tables. The password of the TimesTen cache administration user can be different than the password of the Oracle cache administration user with the same name.
- One or more *cache table owner users* that own the cache tables. You must create a user that owns the TimesTen cache tables that has the same name as the schema owner that owns Oracle Database tables to be cached in the TimesTen database. The graphic shows one cache table owner called `sales` that has the same name as the `sales` schema owner in the Oracle database. The password of a cache table user can be different than the password of the Oracle Database schema owner with the same name.

Use the `ttIsql` utility on the TimesTen instance from an operating system shell or command prompt as the instance administrator, and connect to the `cache1` DSN to create the TimesTen database that is to be used to cache data from an Oracle database:

```
% ttIsql cache1
```

Use `ttIsql` to create a cache administration user on TimesTen. Grant this user the minimum set of privileges required to create cache groups and to perform operations on the cache groups. Any cache administration user that creates, owns, and manages cache groups in the TimesTen database needs at least the `CREATE SESSION`, `CACHE_MANAGER`, and `CREATE ANY TABLE` privileges. In the following example, the cache administration user name is `cacheadmin`, which is the same name as the Oracle Database cache administration user that was created earlier:

```
Command> CREATE USER cacheadmin IDENTIFIED BY ttpwd;
User created.
Command> GRANT CREATE SESSION, CACHE_MANAGER, CREATE ANY TABLE TO cacheadmin;
```

Then, use `ttIsql` to create a cache table owner. In the following example, the cache table owner is `sales`, which is the same name as the schema owner that owns the Oracle database tables that are to be cached. Also, granting `CREATE SESSION` privilege so that `sales` can connect to the TimesTen database, if desired.

```
Command> CREATE USER sales IDENTIFIED BY ttpwd;
User created.
Command> GRANT CREATE SESSION to sales;
Command> exit;
Disconnecting...
Done.
```

Any additional privileges that the cache administration user requires depends on the types of cache groups you create and the operations that you perform on the cache groups. These required privileges are discussed when the cache groups are created.

5

Connect Using an Oracle Wallet with Credentials

For cache operations, you will connect as the cache administration user. You can provide credentials for cache administration users by saving them in an Oracle Wallet, which then can be used for connecting to both the TimesTen and Oracle databases.

The `ttUser -setPwd` command stores any user and password in an Oracle Wallet.

This section describes the process to add cache administration user passwords to an Oracle Wallet.

The following example shows how to use the `ttUser` utility to add both cache administration users to an Oracle Wallet in the `/wallets/cacheadminwallet` directory.

1. If it does not already exist, make a directory for your wallet. This example uses `/wallets` as the directory for the wallet.

   ```
   % mkdir /wallets
   ```

2. Run the `ttUser -setPwd` command to store the TimesTen cache administration user credentials. The `ttUser` utility requires that you provide a subdirectory name that identifies the wallet (since you cannot change the name of an Oracle Wallet). This example provides `cacheadminwallet` as the subdirectory name for the wallet. If `cacheadminwallet` directory does not exist, then the `ttUser` utility creates the `cacheadminwallet` subdirectory and then creates the Oracle Wallet in the `/wallets/cacheadminwallet` directory. The `ttUser` utility prompts for the password for the TimesTen cache administration user `cacheadmin`, which is added to the wallet.

   ```
   % ttUser -setPwd -wallet /wallets/cacheadminwallet -uid cacheadmin
   Enter password:
   ```

3. Run the `ttUser -setOraclePwd` command to store the Oracle cache administration user credentials into the same wallet as the TimesTen cache administration user credentials. The `ttUser` utility prompts for the password for the Oracle cache administration user `cacheadmin`, which is added to the wallet in `/wallets/cacheadminwallet`.

   ```
   % ttUser -setOraclePwd -wallet /wallets/cacheadminwallet -uid cacheadmin
   Enter password:
   ```

Provide the location of the wallet on the connection string with the `PwdWallet` connection attribute. The `UID` connection attribute identifies which credentials to locate within the wallet. The following example provides the cache administration user credentials in an Oracle Wallet when connecting to the database defined in the `cache1` DSN.

```
connect “dsn=cache1;uid=cacheadmin;PwdWallet=/wallets/cacheadminwallet”;
```

For client/server connections, the wallet must exist on the client.

For more details:

- See Providing the Cache Administration User Names and Passwords in an Oracle Wallet in the *Oracle TimesTen In-Memory Database Security Guide* for full details on how to store credentials in an Oracle Wallet.
- See PwdWallet and ttUser in the *Oracle TimesTen In-Memory Database Reference*.

6

Register the Cache Administration User Name and Password

In order to use the database as a cache, you must register the Oracle cache administration user name and password. Connect as the TimesTen cache administration user and call the `ttCacheUidPwdSet` built-in procedure.

This example connects as the `cacheadmin` cache administration user providing credentials in a wallet. After connection, the example calls `ttCacheUidPwdSet` providing the Oracle cache administration user name and password, which registers the Oracle cache administration user name and password within TimesTen.

```
% ttIsql "DSN=cache1;UID=cacheadmin;PwdWallet=/wallets/cacheadminwallet"
Command> call ttCacheUidPwdSet('cacheadmin','cacheadmpwd');
```

7

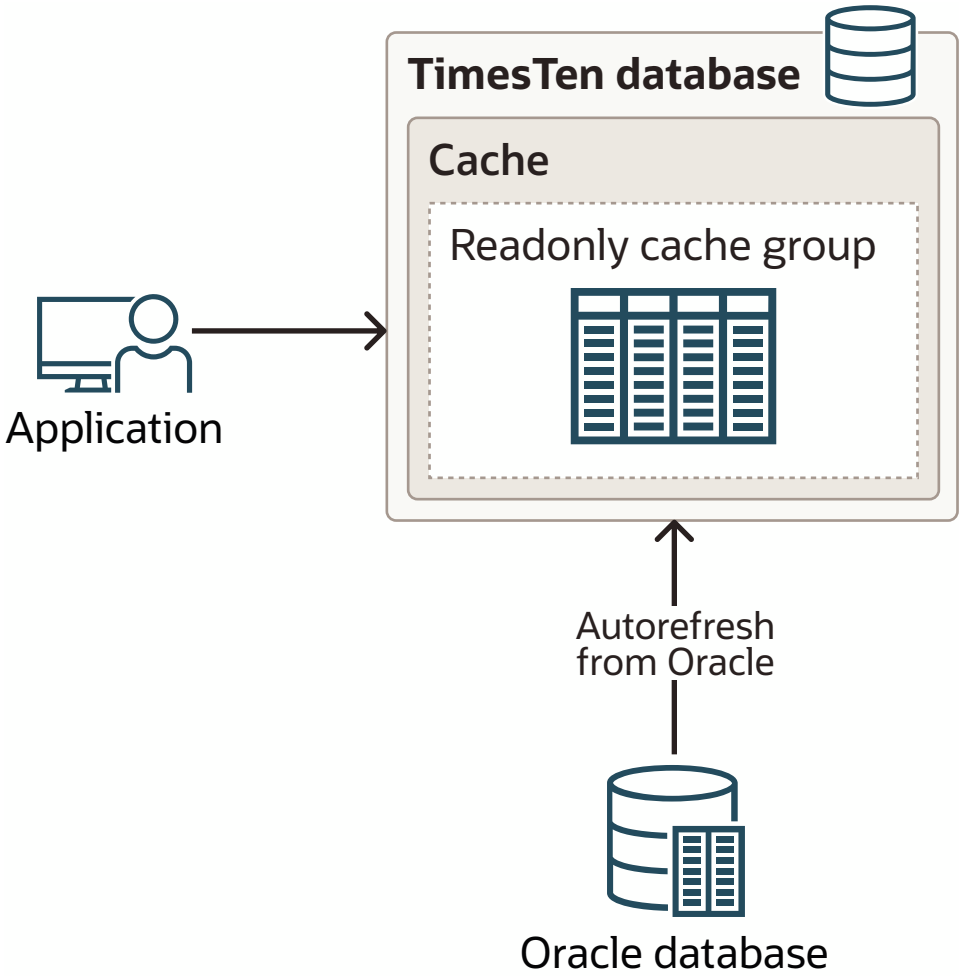
Create Read-Only Cache Groups

Use read-only cache groups when you want to cache read-only data. You create and manage cache groups through standard SQL language in TimesTen.

What is a Read-Only Cache Group?

A read-only cache group enforces a caching behavior in which committed changes on tables in the Oracle database are automatically refreshed to the cache tables in a cache group on the TimesTen database. Using a read-only cache group is suitable for reference data that is heavily accessed by applications.

If a TimesTen database is unavailable for whatever reason, you can still change data within the Oracle database tables that are cached in a read-only cache group. When a TimesTen database returns to operation, changes that were committed on the cached Oracle database tables while the TimesTen database was unavailable are automatically refreshed to the cache tables.



Learn About Read-Only Cache Groups

When you create your cache group, you decide whether your read-only cache group will manually or dynamically load desired cache instances or manually or automatically refresh changes from the Oracle database.

Both static and dynamic read-only cache groups have rules and recommendations for load and refresh operations.

Read-Only Cache Group	Load Operations	Refresh Operations
Static Read-Only cache group	You must perform an initial load with a manual `LOAD CACHE GROUP` SQL statement. A manual load is only used when initially populating the static read-only cache group.	After the initial population with a manual load, all changes to data on the Oracle database are loaded by incremental autorefresh operations, which run at a specified timed interval. The default interval time is 5 minutes. If there is a situation where you want to run a refresh operation in between autorefresh intervals, you can use a manual `REFRESH` SQL statement to synchronize data.

Read-Only Cache Group	Load Operations	Refresh Operations
Dynamic Read-Only cache group	If data does not exist in the cache, then a dynamic load is attempted (for qualifying SQL statements). Once the cache instance is in the cache, any changes are automatically refreshed as normal. A dynamic load request only loads new cache instances that have been inserted into the cache tables on Oracle database. Thus, a dynamic load operation does not load any data if the cache instance currently exists in the cache group. Nornally, there is no manual load performed as the cache group is not initially populated with data.	When using autorefresh (the recommendation) any modifications made to the data on the Oracle database that also exists in the cache group on TimesTen is automatically refreshed (at timed intervals) to the cache group. That is, cache instances that already exist in the cache tables are only changed or deleted with refresh operations. If there is a situation where you want to run a refresh operation in between autorefresh intervals, you can use a manual REFRESH SQL statement to synchronize data.

When to use static or dynamic read-only cache groups:

- Static Read-Only Cache Groups with Autorefresh: Use static read-only cache groups when you want to reduce the latency for read operations using SELECT statements when caching entire tables or static subsets of these tables.

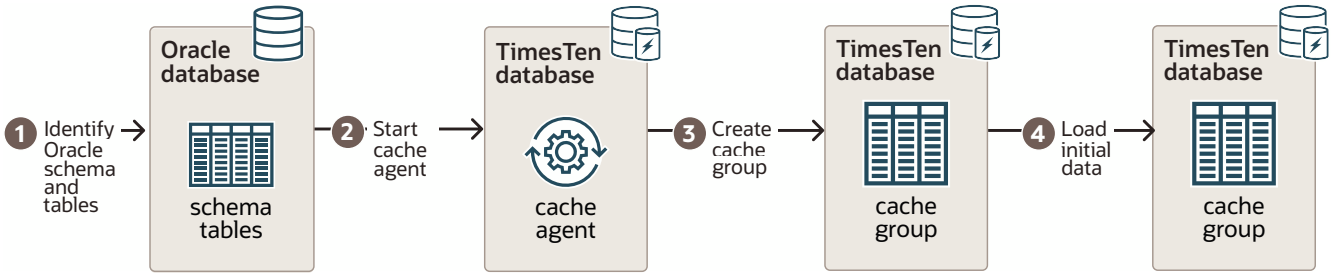
  See Create a Static Read-Only Cache Group with Autorefresh
- Dynamic Read-Only Cache Groups with Autorefresh: Use dynamic read-only cache groups when you are not sure whether you can identify the set of data to be cached in advance and/or whether the potential set of data to be cached can fit in the available memory.

  See Create a Dynamic Read-Only Cache Group with Autorefresh

Create a Static Read-Only Cache Group with Autorefresh

Use static read-only cache groups when you want to cache read-only data where the set of data that needs to be cached can easily be identified and fits within the memory that you allocate to the cache.

To create a static read-only cache group:



These steps are covered in the following sections:

- Task 1: Identify the Schema on the Oracle Database

- Task 2: Start the Cache Agent
- Task 3: Create a Static Read-Only Cache Group on the TimesTen Database
- Task 4: Load Initial Data

After creating the static read-only cache group, you can perform optional tasks to verify that the cache group is performing as expected or to drop the cache groups to start over.

Optional Task	Description
Verify Autorefresh of Static Read-Only Cache Group	Once the static read-only cache group is created and loaded, you can verify the cache group and its data.
Drop the Cache Groups in the TimesTen and Oracle Databases	If you decide that you want to create another cache group in this guide, you can drop the cache group used in this section as it is used in the other sections.

Task 1: Identify the Schema on the Oracle Database

On the Oracle database, use SQL*Plus to connect to the Oracle database as a database administrator.

1. Identify tables to cache on the Oracle Database.

   Since the cache tables are based on the tables you want to cache in the Oracle Database, identify the Oracle Database tables that you want to be cached in the TimesTen database.

   Each table should be either:

   - An Oracle Database table with a primary key on non-nullable columns. The TimesTen cache table primary key must be defined on the full Oracle Database table primary key.
   - An Oracle Database table with non-nullable columns upon which a unique index is defined on one or more of the non-nullable columns in the table. The TimesTen cache table primary key must be defined on all of the columns in the unique index.

   For example, you decide to cache the `sales.customers`, and `sales.orders` tables on the Oracle database. Then, note that the definition of the `customers` and `orders` tables are:

   ```
   CREATE TABLE sales.customers
   (cust_num NUMBER(6) NOT NULL PRIMARY KEY,
    region   VARCHAR2(10),
    name     VARCHAR2(50),
    address  VARCHAR2(100));

   CREATE TABLE sales.orders
   (ord_num      NUMBER(10) NOT NULL PRIMARY KEY,
    cust_num     NUMBER(6) NOT NULL,
    when_placed  DATE NOT NULL,
    when_shipped DATE NOT NULL)
    FOREIGN KEY(cust_num) REFERENCES sales.customers(cust_num));
   ```

2. Since these tables are going to be cached in a read-only cache group, grant the `SELECT` privilege on the `customers` and `orders` tables to the Oracle cache administration user:

   ```
   SQL> GRANT SELECT ON sales.customers TO cacheadmin;
   SQL> GRANT SELECT ON sales.orders TO cacheadmin;
   ```

Task 2: Start the Cache Agent

Start the `ttIsql` utility and connect to the `cache1` DSN as the TimesTen cache administration user, including the TimesTen cache administration user and its credentials in an Oracle Wallet.

One of the most frequently used TimesTen utilities is the `ttIsql` utility. This is an interactive SQL utility that serves the same purpose for TimesTen as SQL*Plus does for Oracle Database.

1. `% ttIsql "DSN=cache1;UID=cacheadmin;PwdWallet=/wallets/cacheadminwallet"`

   See Connect Using an Oracle Wallet with Credentials for directions on how to create an Oracle Wallet.

2. Start the cache agent. The cache agent is a TimesTen daemon process that manages many of the cache-related functions for a TimesTen database.

   ```
   call ttCacheStart;
   ```

Task 3: Create a Static Read-Only Cache Group on the TimesTen Database

Create a static read-only cache group with autorefresh on the TimesTen database.

1. As the TimesTen cache administration user, create a static read-only cache group on a TimesTen database with the `CREATE READONLY CACHE GROUP` SQL statement. Use the unique index columns as the primary key definition.

   The following example shows how the `sales` `customers` and `orders` tables on the Oracle database will be cached in a cache group called `customer_orders`.

   ```
   % ttIsql "DSN=cache1;UID=cacheadmin;PwdWallet=/wallets/cacheadminwallet"
   Command> CREATE READONLY CACHE GROUP customer_orders
   AUTOREFRESH
   FROM sales.customers
    (cust_num NUMBER(6) NOT NULL,
     region   VARCHAR2(10),
     name     VARCHAR2(50),
     PRIMARY KEY(cust_num)),
   sales.orders
    (ord_num      NUMBER(10) NOT NULL,
     cust_num     NUMBER(6) NOT NULL,
     when_placed  DATE NOT NULL,
     when_shipped DATE NOT NULL,
     PRIMARY KEY(ord_num),
     FOREIGN KEY(cust_num) REFERENCES sales.customers(cust_num));
   ```

> **Note**
>
> This SQL statement creates the cache group and the cache tables on the TimesTen database. Autorefresh is added by default for read-only cache groups. Autorefresh defaults to incremental autorefresh running every 5 minutes.
>
> When you choose data types for columns in the TimesTen cache tables, consider the data types of the columns in the Oracle Database tables and choose an equivalent or compatible data type for the columns in the cache tables. See Mappings Between Oracle Database and TimesTen Data Types

2. Exit the `ttIsql` utility that is connected as the TimesTen cache administration user. Then, restart the `ttIsql` utility and connect to the `cache1` DSN as the instance administrator.

   Grant the `SELECT` privilege on the `sales.customers` and `sales.orders` cache tables to the TimesTen cache administration user so that this user can issue `SELECT` queries on these tables.

```
Command> exit;
Disconnecting...
Done.
% ttIsql cache1
Command> GRANT SELECT ON sales.customers TO cacheadmin;
Command> GRANT SELECT ON sales.orders TO cacheadmin;
Command> exit;
Disconnecting...
Done.
```

Task 4: Load Initial Data

Load the cache group.

1. Reconnect as the TimesTen cache administration user. Use the `ttIsql cachegroups` command to view the definition of the `customer_orders` cache group:

```
% ttIsql "DSN=cache1;UID=cacheadmin;PwdWallet=/wallets/cacheadminwallet"
Command> cachegroups;

Cache Group CACHEADMIN.CUSTOMER_ORDERS:

  Cache Group Type: Read Only
  Autorefresh: Yes
  Autorefresh Mode: Incremental
  Autorefresh State: Paused
  Autorefresh Interval: 5 Minutes
  Autorefresh Status: ok
  Aging: No aging defined

  Root Table: SALES.CUSTOMERS
  Table Type: Read Only

  Child Table: SALES.ORDERS
  Table Type: Read Only

1 cache group found.
```

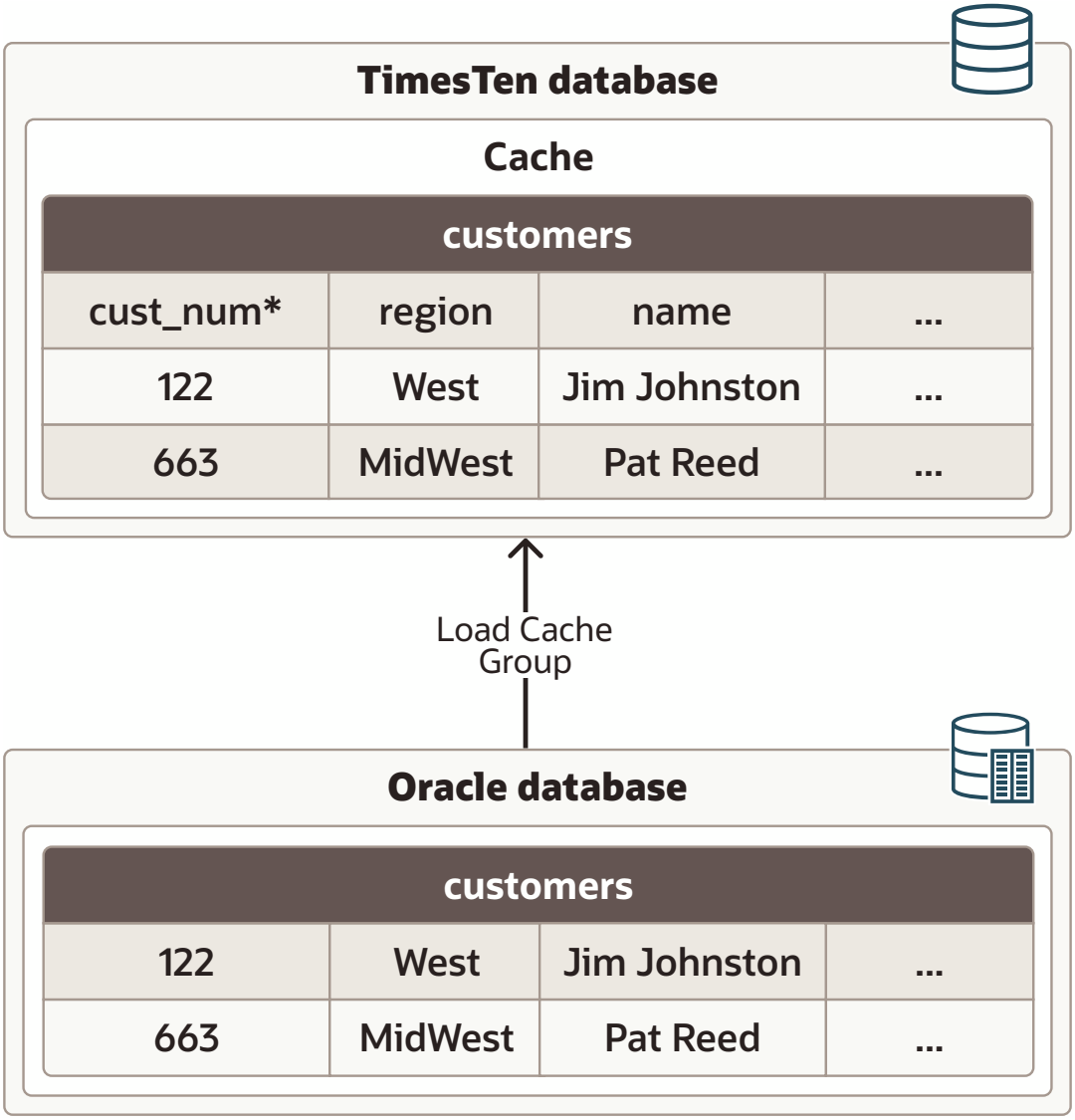
   See ttIsql in the *Oracle TimesTen In-Memory Database Reference*.

   Note that, in the output above, the state of autorefresh operations are currently `Paused` for all of the cache groups that you just created.

```
Autorefresh State: Paused
```

By default, all read-only cache groups defined with incremental autorefresh start as paused (note that the default interval value is 5 minutes).

2. Load the cache group with a `LOAD CACHE GROUP` statement for the first load of the read-only cache group. The `LOAD CACHE GROUP` populates the cache tables (initially, the cache tables are empty) and changes the autorefresh state from `PAUSED` to `ON`.



The following example performs a `LOAD CACHE GROUP` statement for the first load of the `customer_orders` read-only cache group since the cache tables are empty. In this case, the example specifies 3 parallel threads to make the load more efficient and a commit is issued every 256 rows during the load operation.

```
LOAD CACHE GROUP customer_orders COMMIT EVERY 256 ROWS PARALLEL 3;
2 cache instances affected.
```

See LOAD CACHE GROUP in the *Oracle TimesTen In-Memory Database SQL Reference*.

3. Use the `ttIsql` `cachegroups` command again to show that the state for autorefresh state is now `On` for the `customer_orders` cache group:

```
Command> cachegroups;

Cache Group CACHEADMIN.CUSTOMER_ORDERS:

  Cache Group Type: Read Only
  Autorefresh: Yes
  Autorefresh Mode: Incremental
  Autorefresh State: On
  Autorefresh Interval: 5 Minutes
  Autorefresh Status: ok
  Aging: No aging defined

  Root Table: SALES.CUSTOMERS
  Table Type: Read Only

  Child Table: SALES.ORDERS
  Table Type: Read Only

1 cache group found.
```

4. Query the contents of `sales.customers` cache table.

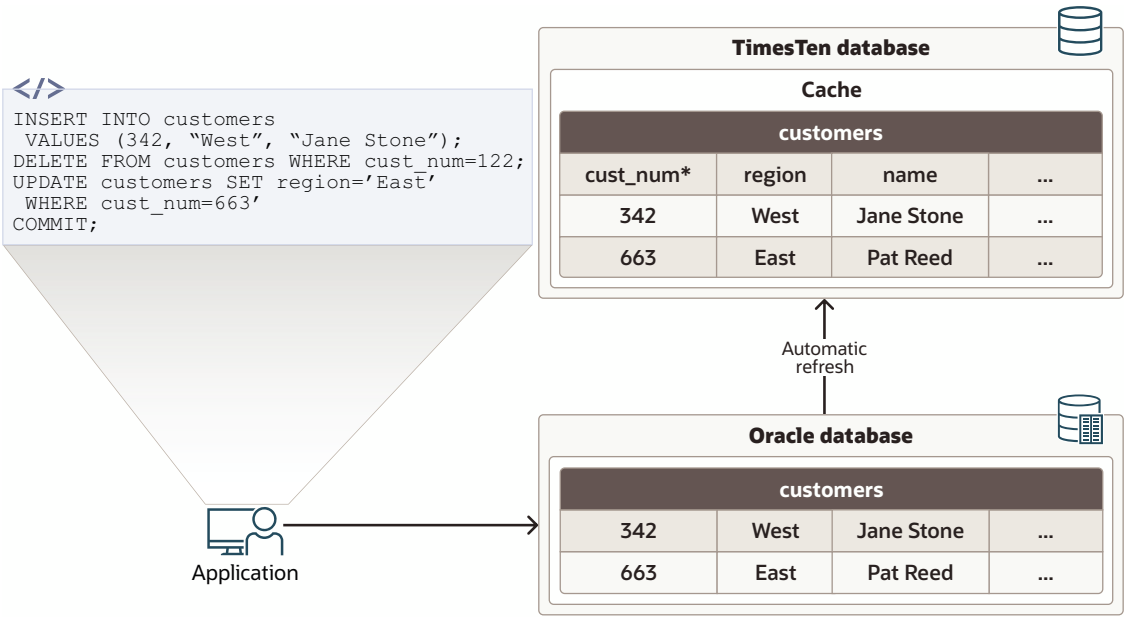
```
Command> SELECT * FROM sales.customers;
122, West, Jim Johnston
663, MidWest, Pat Reed
2 rows found.
```

Since this is a static read-only cache group with autorefresh, all changes on the Oracle database are automatically refreshed into the cache group at the default interval. Since these are read-only cache tables in TimesTen, any DML statements run against them will fail.

Verify Autorefresh of Static Read-Only Cache Group

The following shows how changes are automatically updated to the read-only cache group.

The example in this section inserts a new row, deletes an existing row, updates an existing row in the `customers` cache table, and commits the changes on the Oracle database.



1. On the Oracle database, use SQL*Plus as the Oracle database schema user `sales` to insert a new row, delete an existing row, update an existing row in the Oracle database `customers` table, and commit the changes.

```
SQL> INSERT INTO customers VALUES (342, "West", "Jane Stone");
1 row created.
SQL> DELETE FROM customers WHERE cust_num=122;
1 row deleted.
SQL> UPDATE customers SET region="East" WHERE cust_num=663;
1 row updated.
SQL> COMMIT;
Commit complete.
```

   Since the read-only cache group was created with the default setting for autorefresh with an interval of 5 minutes, the `sales.customers` cache table in the `customer_orders` cache group is automatically refreshed after 5 minutes with the committed changes on the cached Oracle Database `sales.customers` table.

2. On the TimesTen instance as the TimesTen cache administration user, use the `ttIsql` utility to query the contents of the `sales.customers` cache table after the `customer_orders` cache group has been automatically refreshed with the committed changes on the cached Oracle database table:

```
Command> SELECT * FROM sales.customers;
< 342, West, Jane Stone >
< 663, East, Pat Reed >
2 rows found.
Command> exit;
Disconnecting...
Done.
```

Since this is a quick guide on how to create a static read-only cache group, see Overview of Cache Groups and Read-Only Cache Group in the *Oracle TimesTen In-Memory Database Cache Guide* for a more thorough understanding of the concepts behind and the options for a static read-only cache group.

Drop the Cache Groups in the TimesTen and Oracle Databases

All of the examples in this book use the same cache groups. If you want to move on to try other cache group types in this guide, then drop the cache groups (and any cache metadata associated with those cache groups) in both the TimesTen and Oracle databases.

Use the `DROP CACHE GROUP` statement to drop a cache group and its cache tables. On the Oracle database, the metadata objects used to manage the caching of the associated cached tables in the Oracle database are automatically removed.

Use the `ttIsql` utility to connect to the `cache1` DSN as the instance administrator.

Grant the `DROP ANY TABLE` privilege to the TimesTen cache administration user so that this user can drop the underlying cache tables when dropping cache groups.

```
% ttIsql cache1
Command> GRANT DROP ANY TABLE TO cacheadmin;
Command> exit;
Disconnecting...
Done.
```

Start the `ttIsql` utility and connect to the `cache1` DSN as the cache administration user. Use `ttIsql` to drop the `customer_orders` read-only cache group.

```
% ttIsql "DSN=cache1;UID=cacheadmin;PwdWallet=/wallets/cacheadminwallet"
Command> DROP CACHE GROUP customer_orders;
```

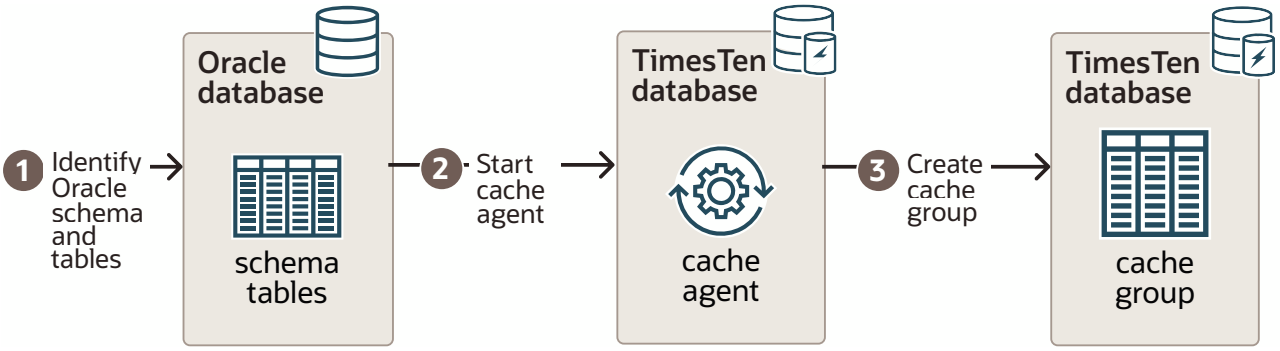
The `customer_orders` cache group and its respective cache tables `sales.customers` and `sales.orders`, are dropped from the TimesTen database. Any metadata objects created for managing the caching operations for this cache group are removed in the Oracle database.

See Dropping a Cache Group in the *Oracle TimesTen In-Memory Database Cache Guide*.

Create a Dynamic Read-Only Cache Group with Autorefresh

To create the dynamic read-only cache group:

The following picture shows the tasks needed when creating a cache group.



These steps are covered in the following sections:

- Task 1: Identify the Schema on the Oracle Database

- Task 2: Start the Cache Agent
- Task 3: Create a Dynamic Read-Only Cache Group on the TimesTen Database

After creating the dynamic read-only cache group, you can perform optional tasks to verify that the cache group is performing as expected or to drop the cache groups to start over.

Optional Task	Description
Verify Dynamic Load with a Dynamic Read-Only Cache Group	Once the dynamic read-only cache group is created and loaded, you can verify the cache group and its data.
Drop the Cache Groups in the TimesTen and Oracle Databases	If you decide that you want to create another cache group in this guide, you can drop the cache group used in this section as it is used in the other sections. See .

Task 1: Identify the Schema on the Oracle Database

On the Oracle database, use SQL*Plus to connect to the Oracle database as a database administrator.

1. Identify tables to cache on the Oracle Database.

   Since the cache tables are based on the tables you want to cache in the Oracle Database, identify the Oracle Database tables that you want to be cached in the TimesTen database.

   Each table should be either:

   - An Oracle Database table with a primary key on non-nullable columns. The TimesTen cache table primary key must be defined on the full Oracle Database table primary key.
   - An Oracle Database table with non-nullable columns upon which a unique index is defined on one or more of the non-nullable columns in the table. The TimesTen cache table primary key must be defined on all of the columns in the unique index.

   For example, you decide to cache the `sales` `customers` and `orders` tables on the Oracle database. Then, note that the definition of the `customers` and `orders` tables are:

   ```
   CREATE TABLE sales.customers
   (cust_num NUMBER(6) NOT NULL PRIMARY KEY,
    region   VARCHAR2(10),
    name     VARCHAR2(50),
    address  VARCHAR2(100));

   CREATE TABLE sales.orders
   (ord_num      NUMBER(10) NOT NULL PRIMARY KEY,
    cust_num     NUMBER(6) NOT NULL,
    when_placed  DATE NOT NULL,
    when_shipped DATE NOT NULL)
    FOREIGN KEY(cust_num) REFERENCES sales.customers(cust_num));
   ```

2. Since these tables are going to be cached in a read-only cache group, grant the `SELECT` privilege on the `customers` and `orders` tables to the Oracle cache administration user:

   ```
   SQL> GRANT SELECT ON sales.customers TO cacheadmin;
   SQL> GRANT SELECT ON sales.orders TO cacheadmin;
   ```

Task 2: Start the Cache Agent

Start the `ttIsql` utility and connect to the `cache1` DSN as the TimesTen cache administration user, including the TimesTen cache administration user and its credentials in an Oracle Wallet.

One of the most frequently used TimesTen utilities is the `ttIsql` utility. This is an interactive SQL utility that serves the same purpose for TimesTen as SQL*Plus does for Oracle Database.

1. `% ttIsql "DSN=cache1;UID=cacheadmin;PwdWallet=/wallets/cacheadminwallet"`

   See Connect Using an Oracle Wallet with Credentials for directions on how to create an Oracle Wallet.

2. Start the cache agent. The cache agent is a TimesTen daemon process that manages many of the cache-related functions for a TimesTen database.

   ```
   call ttCacheStart;
   ```

Task 3: Create a Dynamic Read-Only Cache Group on the TimesTen Database

Create a dynamic read-only cache group with autorefresh on the TimesTen database.

1. Continuing as the TimesTen cache administration user, create a dynamic read-only cache group with autorefresh with the `CREATE DYNAMIC READONLY CACHE GROUP` SQL statement. Use the unique index columns as the primary key definition. Note that autorefresh is configured by default for read-only cache groups.

   ```
   Command> CREATE DYNAMIC READONLY CACHE GROUP customer_orders
   AUTOREFRESH
   FROM sales.customers
    (cust_num NUMBER(6) NOT NULL,
     region   VARCHAR2(10),
     name     VARCHAR2(50),
     PRIMARY KEY(cust_num)),
   sales.orders
    (ord_num      NUMBER(10) NOT NULL,
     cust_num     NUMBER(6) NOT NULL,
     when_placed  DATE NOT NULL,
     when_shipped DATE NOT NULL,
     PRIMARY KEY(ord_num),
     FOREIGN KEY(cust_num) REFERENCES sales.customers(cust_num));
   Command> exit;
   Disconnecting...
   Done.
   ```

> **Note**
>
> This SQL statement creates the cache group and the cache tables on the TimesTen database.
>
> When you choose data types for columns in the TimesTen cache tables, consider the data types of the columns in the Oracle Database tables and choose an equivalent or compatible data type for the columns in the cache tables. See Mappings Between Oracle Database and TimesTen Data Types

2. Start the `ttIsql` utility and connect to the `cache1` DSN as the instance administrator.

   Grant the `SELECT` privilege on the `sales.customers` and `sales.orders` cache tables to the TimesTen cache administration user so that this user can issue a `SELECT` query on this table.

```
% ttIsql cache1
Command> GRANT SELECT ON sales.customers TO cacheadmin;
Command> GRANT SELECT ON sales.orders TO cacheadmin;
Command> exit;
Disconnecting...
Done.
```

Since this is a dynamic read-only cache group with autorefresh, all changes on the Oracle database are automatically refreshed into the cache group at the default interval. You can dynamically request data that does not exist in the cache with a qualified SQL statement. You should not run any DML statements directly against any read-only cache tables in the TimesTen database.

Verify Dynamic Load with a Dynamic Read-Only Cache Group

The following shows how changes are automatically updated to the read-only cache group.

On the Oracle database:

1. Use SQL*Plus as the Oracle database schema user to insert a new row, delete an existing row, update an existing row in the Oracle database `customers` table, and commit the changes.

```
SQL> INSERT INTO customers VALUES (342, "West", "Jane Stone");
1 row created.
SQL> DELETE FROM customers WHERE cust_num=122;
1 row deleted.
SQL> UPDATE customers SET region="East" WHERE cust_num=663;
1 row updated.
SQL> COMMIT;
Commit complete.
```

On the TimesTen database:

Since the dynamic read-only cache group was created specifying autorefresh with the default interval of 5 minutes, the `sales.customers` cache table in the `customer_orders` cache group is automatically refreshed after 5 minutes with the committed changes on the cached Oracle Database `sales.customers` table.

1. As the TimesTen cache administration user, use the `ttIsql` utility to connect to the TimesTen database.

```
% ttIsql "DSN=cache1;UID=cacheadmin;PwdWallet=/wallets/cacheadminwallet"
```

2. Use the `ttIsql` `cachegroups` command to view the definition of the `customer_orders` cache group:

```
Command> cachegroups;
Cache Group CACHEADMIN.CUSTOMER_ORDERS:

  Cache Group Type: Read Only (Dynamic)
  Autorefresh: Yes
  Autorefresh Mode: Incremental
  Autorefresh State: Paused
  Autorefresh Interval: 5 Minutes
  Autorefresh Status: ok
  Aging: LRU on

  Root Table: SALES.CUSTOMERS
  Table Type: Read Only

  Child Table: SALES.ORDERS
  Table Type: Read Only

1 cache group found.
```

   The TimesTen mechanism that captures data changes that occur in the Oracle database and uses those changes to refresh the cached data is called autorefresh. Note that, in the output above, the state of this mechanism is currently `Paused` for the cache group that you just created.

```
Autorefresh State: Paused
```

   By default, all read-only cache groups defined with incremental autorefresh start as paused. Perform a dynamic load request or run a `LOAD CACHE GROUP` statement for the first load of the read-only cache group.

   A dynamic load request or a `LOAD CACHE GROUP` populates the cache tables appropriately (since initially, the cache tables are empty) and changes the autorefresh state from `PAUSED` to `ON`.

3. This example performs a dynamic load request with a qualified SQL query for customer number 342 from the `sales.customers` cache table. This triggers a dynamic load of that cache instance with the committed changes on the cached Oracle database table:

```
Command> SELECT * FROM sales.customers WHERE cust_num=342;
< 342, West, Jane Stone >
1 row found.
```

> **Note**
>
> All changes are automatically refreshed at the autorefresh interval time.

Since this is a quick guide on how to create a dynamic read-only cache group, see Overview of Cache Groups and Dynamic Cache Groups in the *Oracle TimesTen In-Memory Database Cache Guide* for a more thorough understanding of the concepts behind and the options for a dynamic read-only cache group.

Drop the Cache Groups in the TimesTen and Oracle Databases

All of the examples in this book use the same cache groups. If you want to move on to try other cache group types in this guide, then drop the cache groups (and any cache metadata associated with those cache groups) in both the TimesTen and Oracle databases.

Use the `DROP CACHE GROUP` statement to drop a cache group and its cache tables. On the Oracle database, the metadata objects used to manage the caching of the associated cached tables in the Oracle database are automatically removed.

Use the `ttIsql` utility to connect to the `cache1` DSN as the instance administrator.

Grant the `DROP ANY TABLE` privilege to the TimesTen cache administration user so that this user can drop the underlying cache tables when dropping cache groups.

```
% ttIsql cache1
Command> GRANT DROP ANY TABLE TO cacheadmin;
Command> exit;
Disconnecting...
Done.
```

Start the `ttIsql` utility and connect to the `cache1` DSN as the cache administration user. Use `ttIsql` to drop the `customer_orders` read-only cache group.

```
% ttIsql "DSN=cache1;UID=cacheadmin;PwdWallet=/wallets/cacheadminwallet"
Command> DROP CACHE GROUP customer_orders;
```

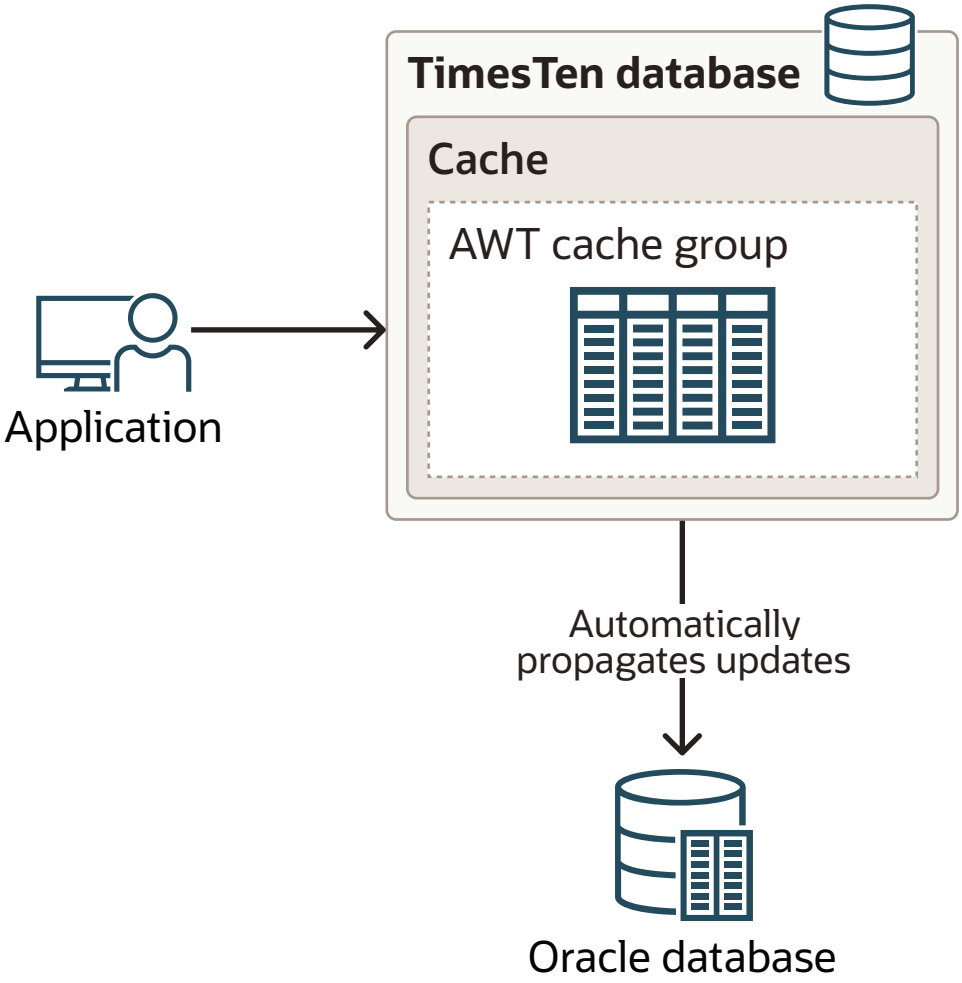
The `customer_orders` cache group and its respective cache tables `sales.customers` and `sales.orders`, are dropped from the TimesTen database. Any metadata objects created for managing the caching operations for this cache group are removed in the Oracle database.

See Dropping a Cache Group in the *Oracle TimesTen In-Memory Database Cache Guide*.

8

Create Static Asynchronous WriteThrough Cache Groups

An Asynchronous WriteThrough (AWT) cache group enforces a caching behavior where committed changes on the TimesTen cache tables are automatically and asynchronously propagated to the cached Oracle Database tables to keep these tables in the two databases synchronized.



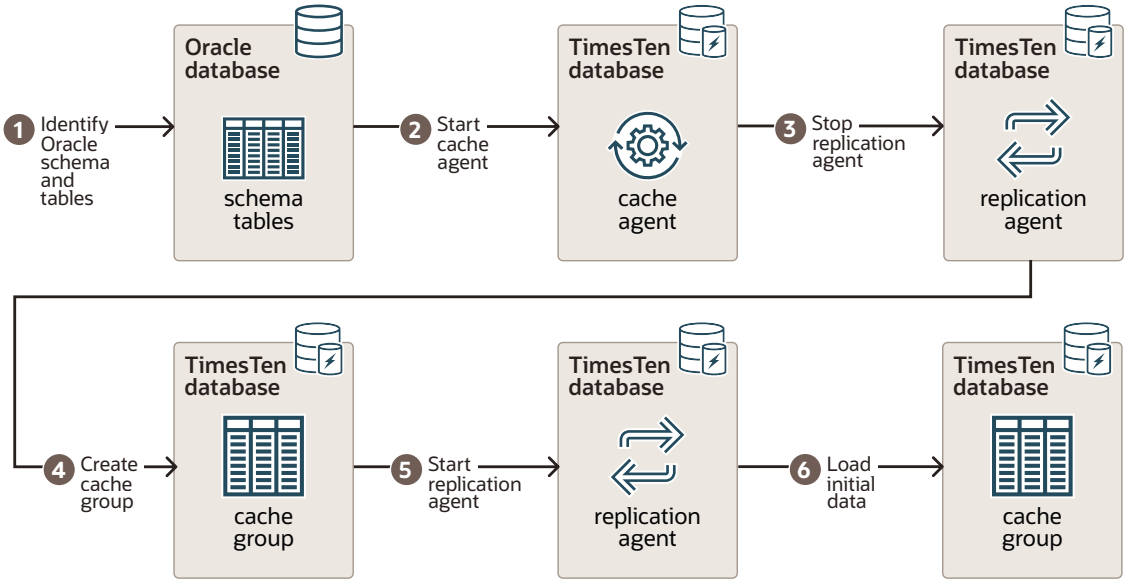
The transaction commit on the TimesTen database occurs asynchronously from the commit on the Oracle database. This enables an application to continue issuing transactions on the TimesTen database without waiting for the Oracle database transactions to complete. When the changes resulting from the transactions in TimesTen are propagated to the Oracle database, transaction integrity and ordering is preserved. Transactions that have committed in TimesTen but have not yet made it to the Oracle database are protected by the TimesTen persistence mechanisms and, optionally, TimesTen high availability (if configured).

Since the AWT cache group uses the replication agent to asynchronously propagate transactions to the Oracle database, these transactions remain in the transaction log buffer and transaction log files until the replication agent confirms they have been fully processed by the Oracle database.

What happens if the data is modified on the Oracle database? Since data automatically flows from TimesTen cache to the Oracle database, any data changed in the cached Oracle database tables is not automatically propagated to the TimesTen cache. Changes propagated from the cache may overwrite changes made directly in the Oracle database. For tables that are cached using AWT, TimesTen should be considered the primary source where the application makes changes to the data. It is possible to refresh the AWT cache tables in TimesTen by using the `REFRESH CACHE GROUP` statement, which discards the table contents in TimesTen and repopulates the tables from the Oracle database.

Create Static Asynchronous WriteThrough Cache Group

The following graphic shows the steps necessary to create a static AWT cache group:



These steps are covered in the following sections:

- Task 1: Identify the Schema on the Oracle Database
- Task 2: Start the Cache Agent
- Task 3: Stop the Replication Agent
- Task 4: Create the Static AWT Cache Group on the TimesTen Database
- Task 5: Start the Replication Agent
- Task 6: Load Initial Data

After creating the static AWT cache group, you can perform optional tasks to verify that the cache group is performing as expected or to drop the cache groups to start over.

Optional Task	Description
Verify Static AWT Cache Group	Once the AWT cache group is created and loaded, you can verify the cache group and its data.

Optional Task	Description
Drop the Cache Groups in the TimesTen and Oracle Databases	If you decide that you want to create another cache group in this guide, you can drop the cache group used in this section as it is used in the other sections.

Task 1: Identify the Schema on the Oracle Database

On the Oracle database, use SQL*Plus to:

1. Identify Oracle Database tables to cache.

   Since the cache tables are based on the tables you want to cache in the Oracle Database, identify the Oracle Database tables that are to be cached.

   Each table should be either:

   - An Oracle Database table with a primary key on non-nullable columns. The TimesTen cache table primary key must be defined on the full Oracle Database table primary key.
   - An Oracle Database table with non-nullable columns upon which a unique index is defined on one or more of the non-nullable columns in the table. The TimesTen cache table primary key must be defined on all of the columns in the unique index.

   For example, you decide to cache the `sales.customers` table on the Oracle database. Then, note that the definition of the `customers` tables is:

   ```
   CREATE TABLE sales.customers
   (cust_num NUMBER(6) NOT NULL PRIMARY KEY,
    region   VARCHAR2(10),
    name     VARCHAR2(50),
    address  VARCHAR2(100));
   Table created.
   ```

2. Since these tables are going to be cached in a read-write cache group, grant the `SELECT`, `INSERT`, `UPDATE` and `DELETE` privileges on the `customer` table to the Oracle cache administration user:

   ```
   SQL> GRANT SELECT, INSERT, UPDATE, DELETE ON sales.customers TO cacheadmin;
   ```

Task 2: Start the Cache Agent

Start the `ttIsql` utility and connect to the `cache1` DSN as the TimesTen cache administration user, including the TimesTen cache administration user and its credentials in an Oracle Wallet.

One of the most frequently used TimesTen utilities is the `ttIsql` utility. This is an interactive SQL utility that serves the same purpose for TimesTen as SQL*Plus does for Oracle Database.

1. ```
   % ttIsql "DSN=cache1;UID=cacheadmin;PwdWallet=/wallets/cacheadminwallet"
   ```

   See Connect Using an Oracle Wallet with Credentials for directions on how to create an Oracle Wallet.

2. Start the cache agent. The cache agent is a TimesTen daemon process that manages many of the cache-related functions for a TimesTen database.

   ```
   call ttCacheStart;
   ```
   ```

Task 3: Stop the Replication Agent

Stop the replication agent. The replication agent is a TimesTen daemon process that propagates committed changes on TimesTen cache tables in AWT cache groups to the cached Oracle Database tables.

```
CALL ttRepStop;
```

Task 4: Create the Static AWT Cache Group on the TimesTen Database

Create a static AWT cache group.

As the TimesTen cache administration user, create a static AWT cache group with the `CREATE ASYNCHRONOUS WRITETHROUGH CACHE GROUP` SQL statement. Use the unique index columns as the primary key definition.

The following statement creates an AWT cache group `awt_customers` that caches the `sales.customers` table:

```
CREATE ASYNCHRONOUS WRITETHROUGH CACHE GROUP awt_customers
FROM sales.customers
 (cust_num NUMBER(6) NOT NULL,
  region   VARCHAR2(10),
  name     VARCHAR2(50),
  PRIMARY KEY(cust_num));
```

> **Note**
>
> This SQL statement creates the cache group and the cache tables on the TimesTen database.
>
> When you choose data types for columns in the TimesTen cache tables, consider the data types of the columns in the Oracle Database tables and choose an equivalent or compatible data type for the columns in the cache tables. See Mappings Between Oracle Database and TimesTen Data Types

Task 5: Start the Replication Agent

Performing asynchronous writethrough operations requires that the replication agent be running on the TimesTen database that contains AWT cache groups.

Running a `CREATE ASYNCHRONOUS WRITETHROUGH CACHE GROUP` statement creates a replication scheme that enables committed changes on the TimesTen cache tables to be asynchronously propagated to the cached Oracle Database tables.

After you have created AWT cache groups, start the replication agent on the TimesTen database by calling the `ttRepStart` built-in procedure as the cache administration user.

> **Note**
>
> The `ttRepStart` connects to the Oracle database using the credentials that were registered with the `ttCacheUidPwdSet` built-in procedure back in Register the Cache Administration User Name and Password. Since you set `CacheAdminWallet=1` in the DSN, the credentials are passed within an Oracle Wallet that contains the cache administration user name and the passwords for both cache administration users.

After you have created an AWT cache group, start the replication agent on the TimesTen database. Exit ttIsql.

```
Command> CALL ttRepStart;
Command> exit;
Disconnecting...
Done.
```

Task 6: Load Initial Data

Load the cache group.

1. Start the `ttIsql` utility and connect to the `cache1` DSN as the instance administrator.

   Grant the `SELECT`, `INSERT`, `UPDATE` and `DELETE` privileges on the `sales.customers` cache table to the TimesTen cache administration user so that this user can issue `SELECT`, `INSERT`, `UPDATE` and `DELETE` SQL statements on this table. The `INSERT`, `UPDATE` and `DELETE` privileges on the `sales.customers` table are required to run write through operations from the TimesTen cache table to the cached Oracle Database table.

   ```
   % ttIsql cache1
   Command> GRANT SELECT, INSERT, UPDATE, DELETE ON sales.customers TO cacheadmin;
   Command> exit;
   Disconnecting...
   Done.
   ```

2. Start the `ttIsql` utility and connect to the `cache1` DSN as the TimesTen cache administration user.

   Perform a `LOAD CACHE GROUP` statement for the first load of the AWT cache group since the cache tables are empty.

   ```
   % ttIsql "DSN=cache1;UID=cacheadmin;PwdWallet=/wallets/cacheadminwallet"
   Command> LOAD CACHE GROUP awt_customers COMMIT EVERY 256 ROWS PARALLEL 3;
   2 cache instances affected.
   ```

3. Query the contents of `sales.customer` cache table.

   ```
   Command> SELECT * FROM sales.customers;
   < 122 West     Jim Johnston >
   < 663 MidWest  Pat Reed >
   2 rows found.
   ```

4. Use the `ttIsql` `cachegroups` command to view the definition of the `awt_customers` cache group:

   ```
   Command> cachegroups;

   Cache Group CACHEADMIN.AWT_CUSTOMERS:
   ```

```
    Cache Group Type: Asynchronous Writethrough
    Autorefresh: No
    Aging: LRU on

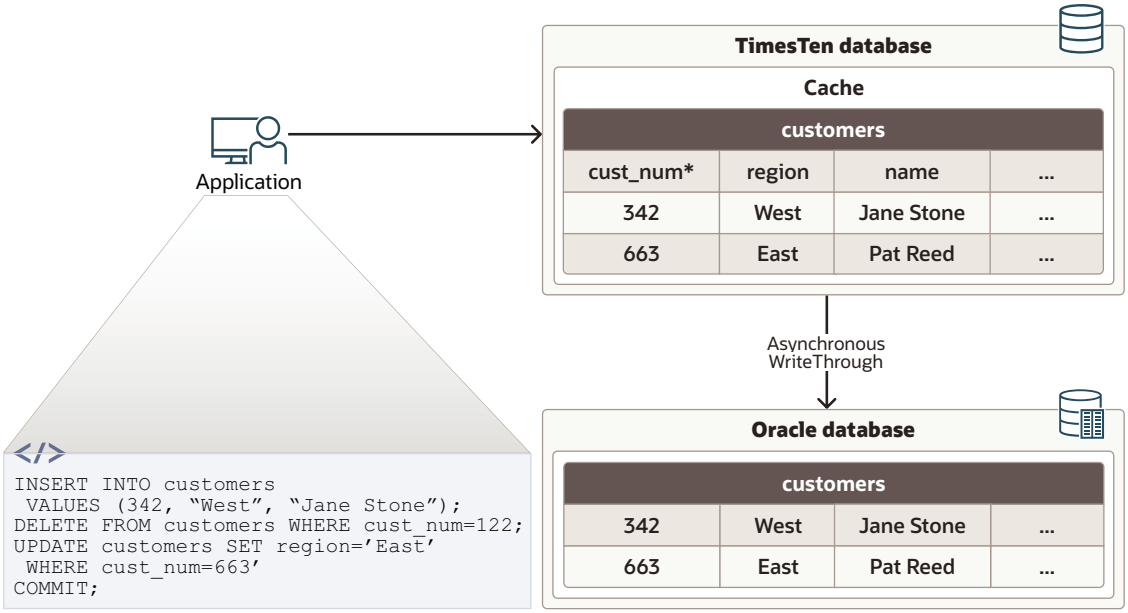
    Root Table: SALES.CUSTOMERS
    Table Type: Propagate

1 cache group found.
```

Verify Static AWT Cache Group

Since this is a static AWT cache group, all changes made on the TimesTen database are automatically propagated to the Oracle database.

The following example inserts a new row, deletes an existing row, updates an existing row in the `customers` cache table, and commits the changes on the TimesTen database.



Connect to ttIsql as `sales`, then:

```
Command> INSERT INTO customers VALUES (342, "West", "Jane Stone");
1 row created.
Command> DELETE FROM customers WHERE cust_num=122;
1 row deleted.
Command> UPDATE customers SET region="East" WHERE cust_num=663;
1 row updated.
Command> COMMIT;
Command> exit;
Disconnecting...
Done.
```

You can verify this by connecting to the Oracle database. As the Oracle database schema user `sales`, use SQL*Plus to query the contents of the `customer` table:

```
SQL> SELECT * FROM customers;
cust_num  region  name
----------------------------------------
```

```
342       West    Jane Stone
663       East    Pat Reed
```

Since this is a quick guide on how to create a static AWT cache group, see Overview of Cache Groups and Asynchronous WriteThrough (AWT) Cache Group in the *Oracle TimesTen In-Memory Database Cache Guide* for a more thorough understanding of the concepts behind and the options for a static AWT cache group.

Drop the Cache Groups in the TimesTen and Oracle Databases

All of the examples in this book use the same cache groups. If you want to move on to try other cache group types in this guide, then drop the cache groups (and any cache metadata associated with those cache groups) in both the TimesTen and Oracle databases.

1. Stop the Replication Agent.
2. Drop the Cache Groups.

Stop the Replication Agent

As the TimesTen cache administration user, use the `ttIsql` utility to call the `ttRepStop` built-in procedure to stop the replication agent on the TimesTen database.

```
Command> call ttRepStop;
Command> exit
Disconnecting...
Done.
```

Drop the Cache Groups

Use the `DROP CACHE GROUP` statement to drop a cache group and its cache tables. On the Oracle database, the metadata objects used to manage the caching of the associated cached tables in the Oracle database are automatically removed.

Use the `ttIsql` utility to connect to the `cache1` DSN as the instance administrator.

Grant the `DROP ANY TABLE` privilege to the TimesTen cache administration user so that this user can drop the underlying cache tables when dropping cache groups.

```
% ttIsql cache1
Command> GRANT DROP ANY TABLE TO cacheadmin;
Command> exit
Disconnecting...
Done.
```

Start the `ttIsql` utility and connect to the `cache1` DSN as the cache administration user. Use `ttIsql` to drop the `customer_orders` AWT cache group.

```
% ttIsql "DSN=cache1;UID=cacheadmin;PwdWallet=/wallets/cacheadminwallet"
Command> DROP CACHE GROUP awt_customers;
```

The `awt_customers` cache group and its respective cache table `sales.customers` are dropped from the TimesTen database. Any metadata objects created for managing the caching operations for this cache group are removed in the Oracle database.

See Dropping a Cache Group in the *Oracle TimesTen In-Memory Database Cache Guide*.

9 Next Steps

Here are some links for further investigation now that you have a basic understanding of cache concepts, cache groups, and some of the cache operations.

Task	Description	More Information
Try the LiveLab for cache	There is a LiveLab that you can use to become more familiar with how these steps work. There is a slight difference in the lab in that the lab assigns a small tablespace due to the fact that this is a lab and not a production environment.	Accelerate your Applications - Achieve Blazing Fast SQL With an Oracle TimesTen Cache.
Discover all of the available cache group types.	There are more than just two types of cache groups. This quick start guide showed the read-only and asynchronous writethrough cache groups.	Cache Group Types
Familiarize yourself with all autorefresh details.	You were briefly introduced to autorefresh when using the defaults. There are multiple ways you can manage how autorefresh works.	Automatically Refreshing a Cache Group
Read about all methods for synchronizing data between Oracle and TimesTen databases.	There are several methods for loading data from an Oracle database to the TimesTen database. There are a couple of methods for propagating data from the TimesTen database to the Oracle database.	Transmitting Changes Between the TimesTen and Oracle Databases
Learn how to specify what rows are cached in the TimesTen database.	Several cache operation SQL statements can contain a `WHERE` clause to restrict the rows to cache in the TimesTen database for particular cache group types. There are full details about when and how to use a `WHERE` clause.	Using a WHERE Clause
Familiarize yourself with all dynamic load details.	You were introduced to dynamic load. You specify whether your cache group is dynamically loaded by specifying the `DYNAMIC` keyword during cache group definition. There are more details about how to use dynamic load.	Dynamic Cache Groups

Task	Description	More Information
Learn how to pass SQL statements through to the Oracle database while connected to a TimesTen database.	All of the examples in this guide show the user logging into Oracle database to perform operations. However, there is a method called passthrough where when the application runs SQL statements on a TimesTen connection, the SQL statement is performed either in the TimesTen database or passed through to the Oracle database. Whether the SQL statement is performed in the TimesTen or Oracle database depends on the composition of the statement and the setting of the `PassThrough` connection attribute.	Setting a Passthrough Level
Manage your cache environment.	Once you have your cache groups up and running, you can manage your cache operations.	Managing a Caching Environment
Clean up your cache environment.	This section has information on how to clean up your cache environment.	Cleaning Up the Caching Environment