# Coherence and GoldenGate HotCache

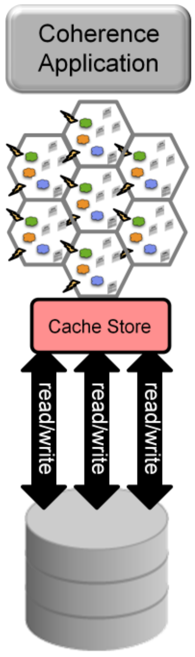
## Coherence and Databases

- Coherence is designed to be a highly scalable, efficient in-memory cache.
- Databases are designed to support large amounts of persistent data.
- Coherence can use components known as CacheStores to persist data to a database.
- Such solutions work best when Coherence is the system of record.

What happens when you need persistent storage, but such storage is also accessed outside Coherence?

## Database-Backed Coherence

- Scenario:
  - Coherence as the system of record
- Benefits:
  - It is simple.
  - CacheStore implementations can be customized as required.
  - Stock CacheStores exist for many situations.
- Drawbacks:
  - External system access can cause stale data.

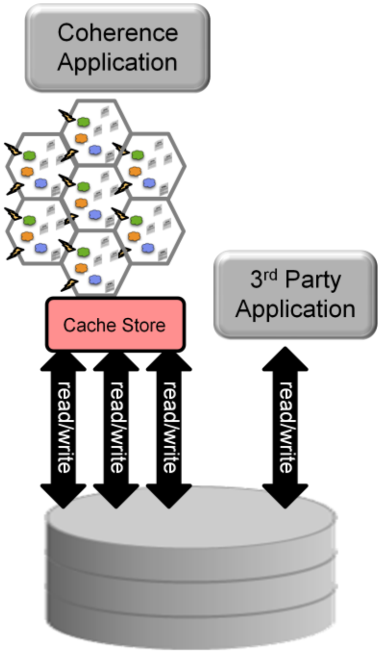


12 - 3

One of the simplest ways to provide persistent storage with Coherence is via the use of a CacheStore. A CacheStore is a configured software component that intercepts reads and writes, and forwards them to a database. Many mechanisms exist for configuring a CacheStore. However, a solution in which the backing database is not accessed by any other software would be the best solution

## Shared Database as System of Record

- Scenario:
  - Coherence backed by database
  - External applications accessing database
- Benefits
  - Simple
  - Works if Coherence is source of all database changes
- Drawbacks
  - Potential stale data



12 - 4

The shared database scenario builds on the simple scenario where Coherence is the system of record, but uses a database as a backing store. The shared database scenario adds an additional component, the third-party application that performs read/write access against the database. The out-of-the-box solutions to this problem of external application access provide varying degrees of success. Solutions might include:

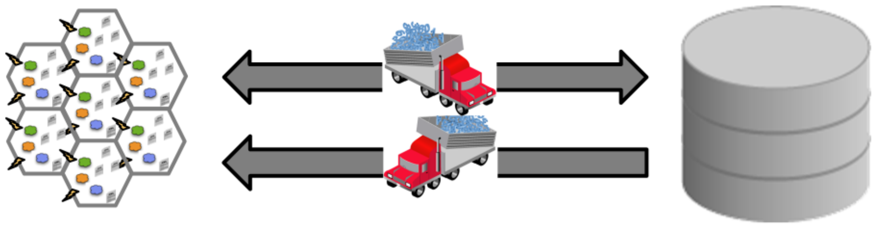
- **Cache Expiry:** Setting a low value for cache data causes it to expire and be re-read. However, such a solution is inefficient because it treats all data similarly and is likely to expire fresh data.
- **Refresh Ahead:** Refresh ahead re-reads data regularly but may be subject to long read latency as well as re-reading fresh data.

Each of the out-of-the-box solutions also comes with drawbacks. For example, Coherence data might be stale or out-of-date for a period of time when a third-party database write occurs. In addition, Coherence may expire perfectly valid data, resulting in unnecessary read activity. Each of these problems is solved by the use of the GoldenGate HotCache adapter.

## GoldenGate HotCache

Oracle GoldenGate HotCache:

- Solves the database-to-Coherence stale data problem
- Leverages existing technologies, including:
  - TopLink Grid, Java Persistence API, and GoldenGate
- Is event-driven and has low latency
- Requires little or no application changes

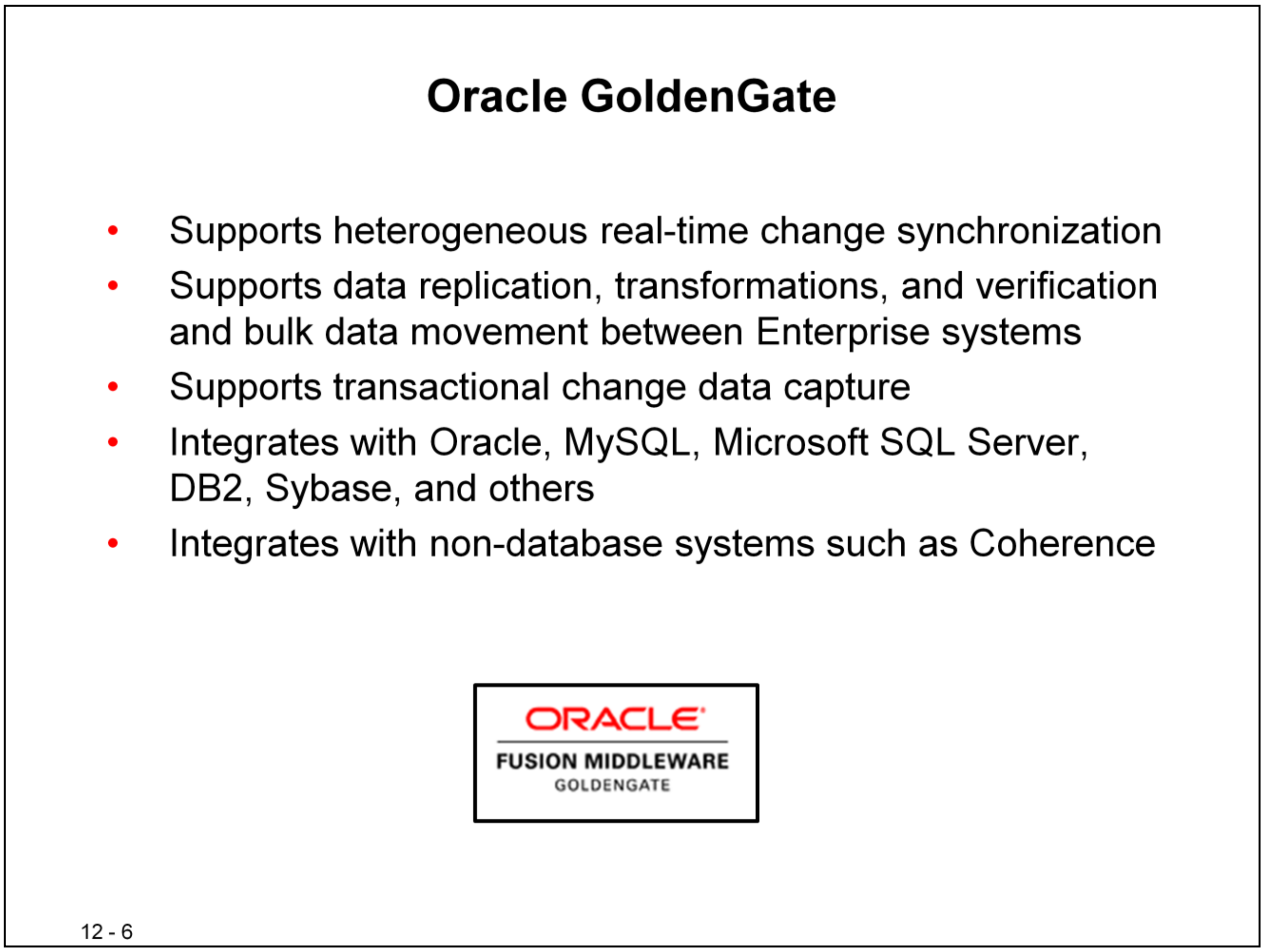


Using JPA and a custom cache loader

12 - 5

GoldenGate is designed to replicate and transform data between databases in real time. The GoldenGate HotCache adapter is designed to provide similar GoldenGate-style capabilities but between Coherence and a backing store—typically a database, but in reality a supported GoldenGate platform.

GoldenGate can be used to replace the Coherence write-behind strategy, and removes many of its limitations in terms of delay, potential loss of transactions, and performance.



## Oracle GoldenGate

- Supports heterogeneous real-time change synchronization
- Supports data replication, transformations, and verification and bulk data movement between Enterprise systems
- Supports transactional change data capture
- Integrates with Oracle, MySQL, Microsoft SQL Server, DB2, Sybase, and others
- Integrates with non-database systems such as Coherence

ORACLE
FUSION MIDDLEWARE
GOLDENGATE

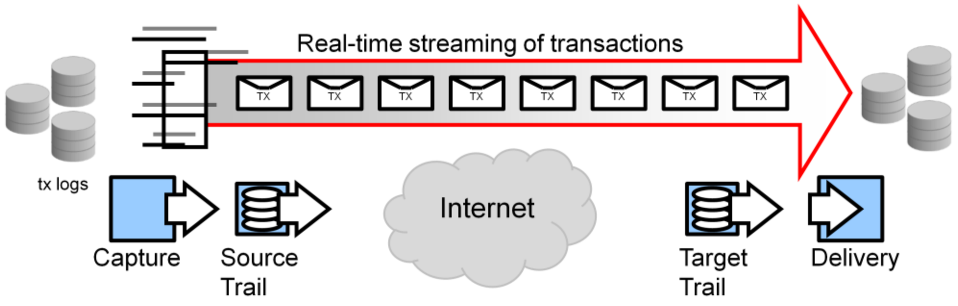
12 - 6

Oracle GoldenGate is designed as a gateway between databases and similar data-drive products. GoldenGate primarily acts as a synchronization engine, allowing changes in one database to be moved seamlessly to another. However, GoldenGate is not limited to one-to-one relationships but rather supports one-to-many and a variety of other combinations. It moves beyond Oracle databases, and can synchronize with other systems such as Microsoft SQL Server, DB3, Sybase, MySQL, and others, including Oracle Coherence. In addition to its core capabilities, Oracle GoldenGate can perform a variety of transformations and bulk data movement, and is completely transaction-aware.

## Oracle GoldenGate

Oracle GoldenGate provides real-time:

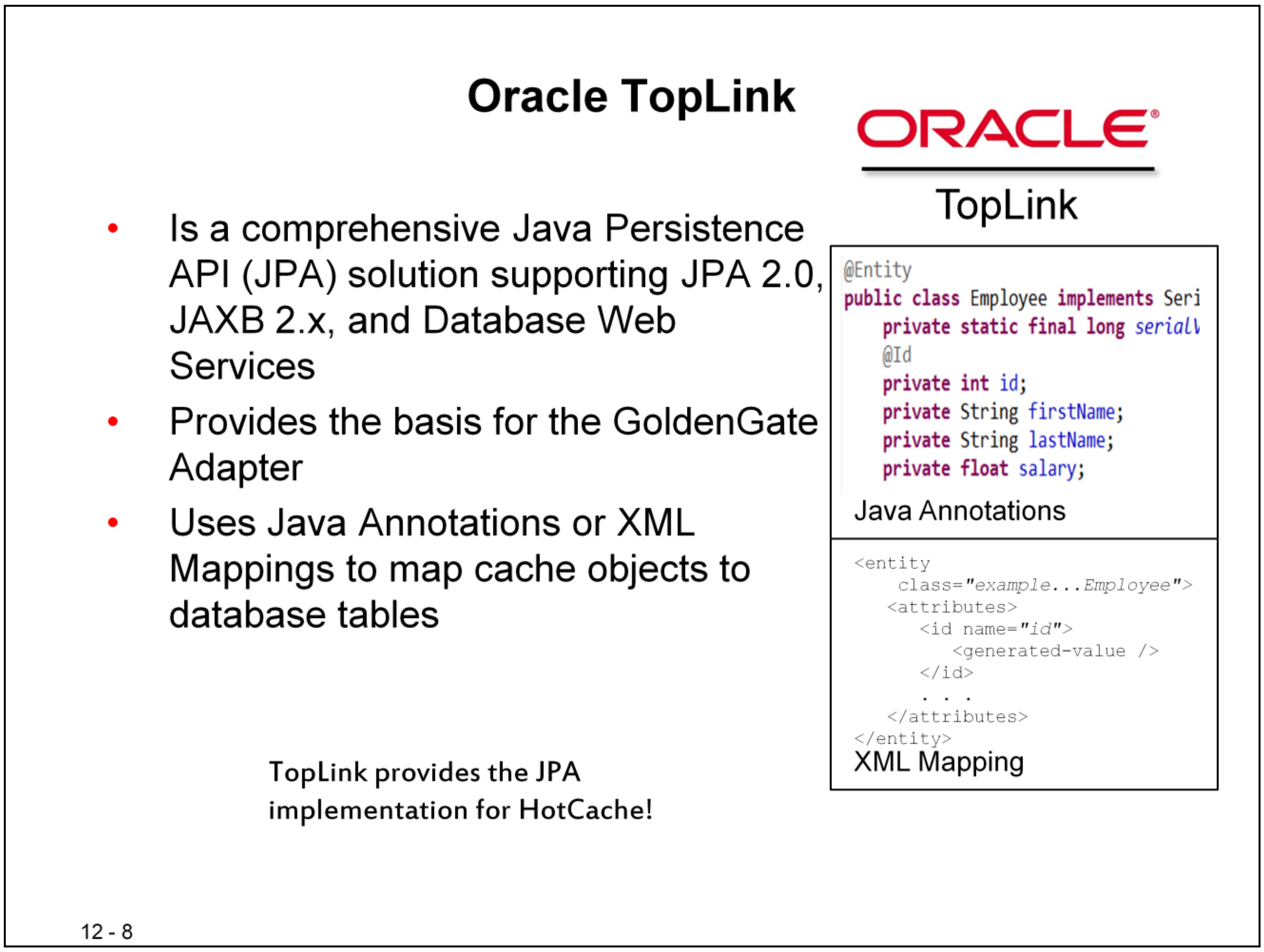
- Capture: Capture data from Oracle and other database vendors as it happens.
- Transformation: Transform captured data as required.
- Distribute: Transfer changes to one or more systems.
- Deliver: Deliver transactions across heterogeneous systems, keeping all systems synchronized.



12 - 7

For real-time data warehousing environments, Oracle GoldenGate captures and delivers changed data to the data warehouse or operational data store in real time. Because this is log-based, there is minimal impact on the source; there are no batch windows; and it moves the new transactions in a source system in subseconds. During the movement phase, each transaction's commit boundaries are maintained to ensure data integrity. ODI performs complex transformations within the database for maximum performance.

The other benefit of this approach is data recoverability in case there is an outage during data movement. The Oracle GoldenGate trail files that store the changed data are persistent, so they can be re-applied (if needed) to the target and also to the source system.



## Oracle TopLink

- Is a comprehensive Java Persistence API (JPA) solution supporting JPA 2.0, JAXB 2.x, and Database Web Services
- Provides the basis for the GoldenGate Adapter
- Uses Java Annotations or XML Mappings to map cache objects to database tables

TopLink

```
@Entity
public class Employee implements Seri
    private static final long serialV
    @Id
    private int id;
    private String firstName;
    private String lastName;
    private float salary;
```

Java Annotations

```
<entity
    class="example...Employee">
   <attributes>
      <id name="id">
         <generated-value />
      </id>
      . . .
   </attributes>
</entity>
```

XML Mapping

TopLink provides the JPA implementation for HotCache!

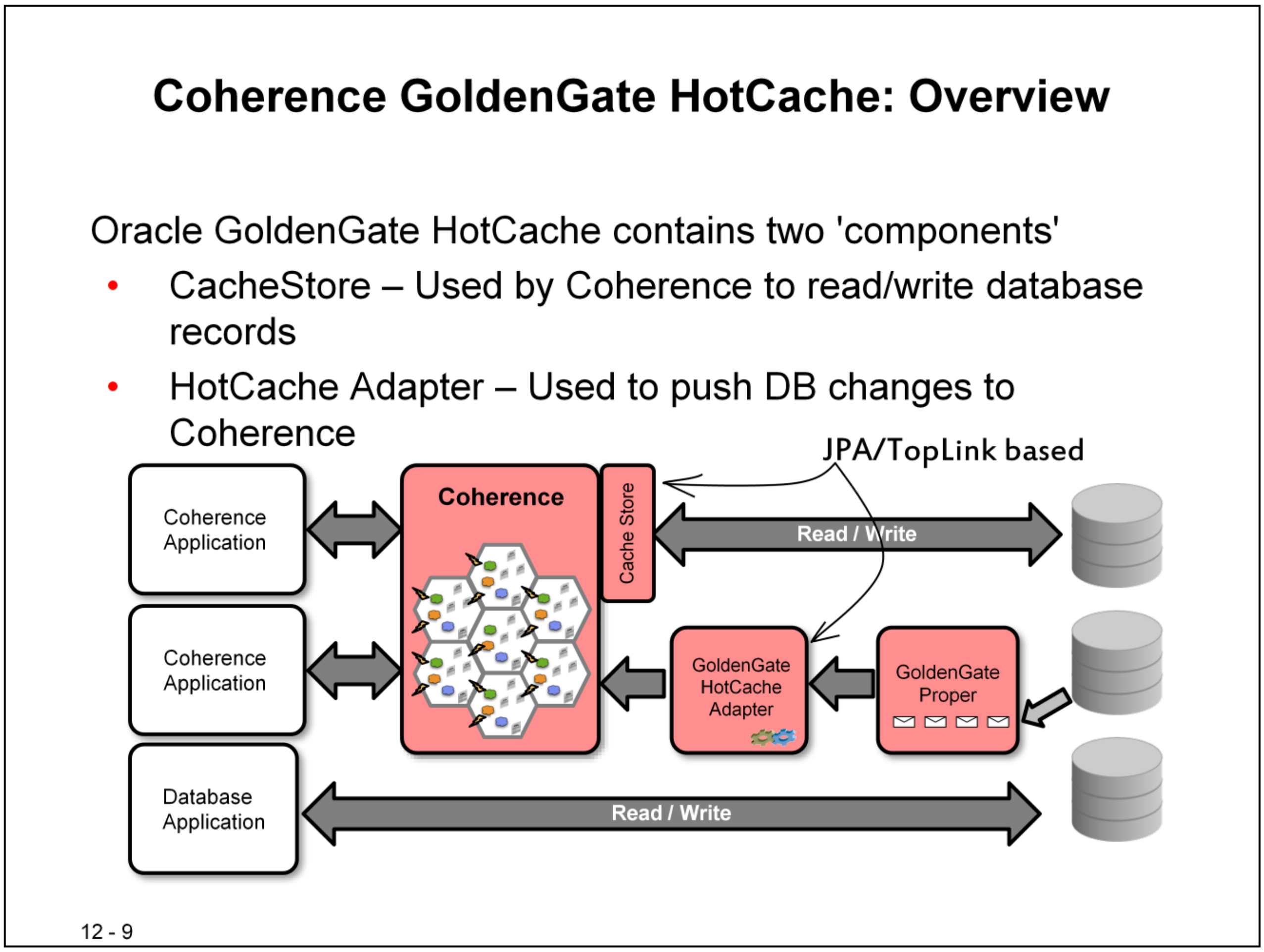
12 - 8

TopLink is Oracle’s strategic persistence framework for Java. TopLink supports a number of different persistence standards, including:

- Java Persistence API (JPA)
- Java Architecture for XML Binding (JAXB)
- A new component called DBWS that allows you to access relational database through web services. It is an XML/Relational bridge technology that is wrapped in a web service.

TopLink is based on the open source EclipseLink project that was developed at the Eclipse Foundation. Oracle augments EclipseLink with support for Coherence integration and distributes this combination as Oracle TopLink. The Coherence integration feature is referred to as TopLink Grid.

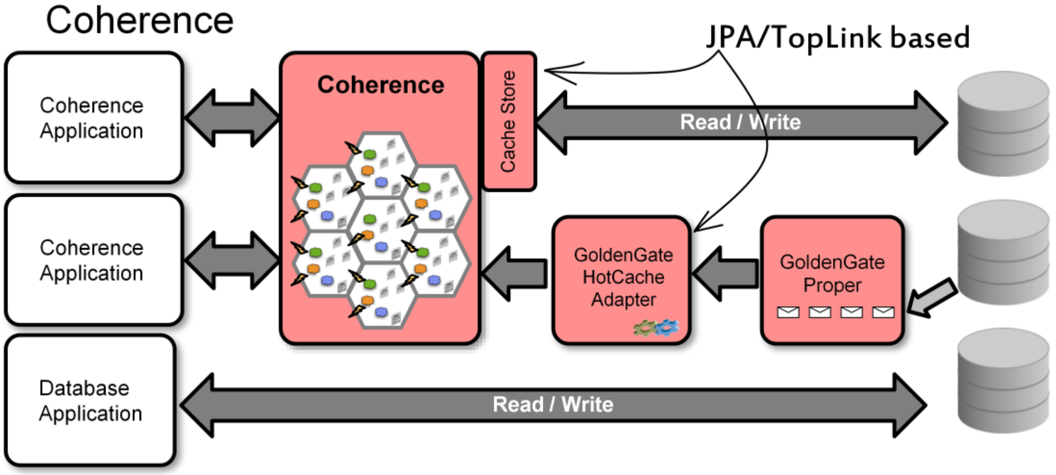
HotCache uses TopLink to provide the mapping between Coherence and the database. Using JPA, either via Java annotations or XML Mapping, Coherence objects are mapped to their respective database elements.



## Coherence GoldenGate HotCache: Overview

Oracle GoldenGate HotCache contains two 'components'

- CacheStore – Used by Coherence to read/write database records
- HotCache Adapter – Used to push DB changes to Coherence



12 - 9

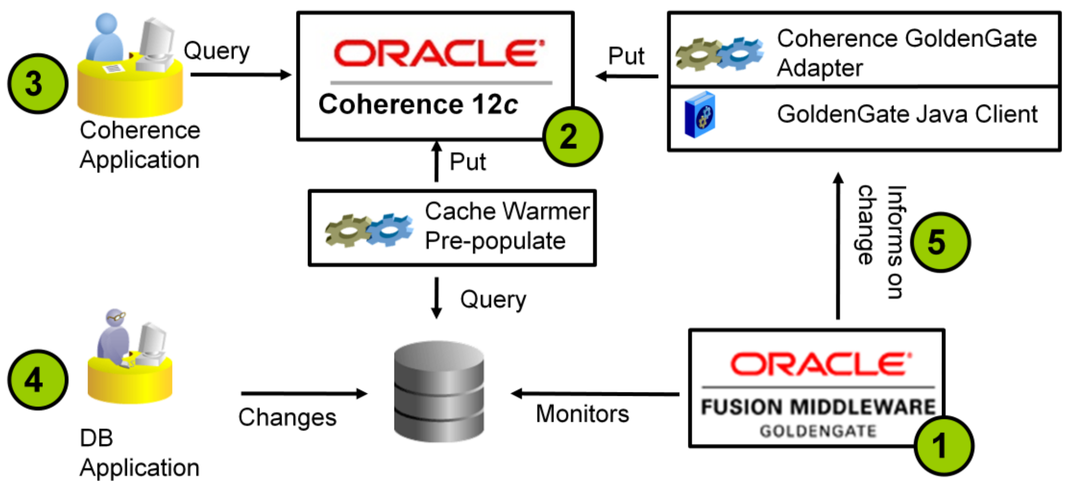
GoldenGate is designed to replicate and transform data between databases in real time. The GoldenGate HotCache adapter is designed to provide similar GoldenGate-style capabilities but between Coherence and a backing store—typically a database, but in reality any supported GoldenGate platform.

GoldenGate and Coherence interact in two distinct ways:

- **Coherence CacheStore:** A Coherence cache store is registered with Coherence and tracks read and write changes to the database. In a Coherence/Database environment where only Coherence interacts with and updates the database, this solution would be sufficient. However, if an external mechanism can modify the database, a cache store would be insufficient because the Coherence data might become stale.
- **GoldenGate:** GoldenGate works as it traditionally does by interacting with a database and capturing the changes generated by third party or external applications. These changes are captured in GoldenGate trail files. The GoldenGate HotCache adapter is then notified of the changes and these changes are pushed upstream to Coherence, resolving any external updates. Using this mechanism, GoldenGate and the database behind it become an event source, providing real-time data change events to Coherence.

The combination of Coherence and GoldenGate provides an efficient, low-latency solution to the general problem of multiple applications accessing and changing both Coherence and Database data.

## Coherence GoldenGate Adapter



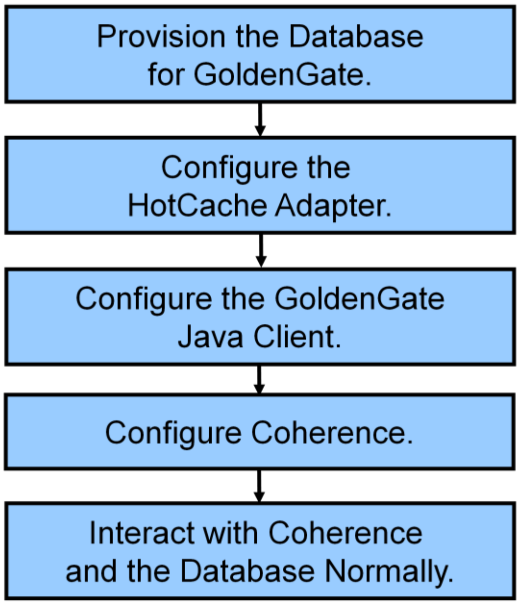
12 - 10

The Coherence GoldenGate Adapter interacts with GoldenGate by registering and receiving changes to the database. The diagram describes the process at a high level.

1. GoldenGate and the Coherence GoldenGate Adapter are started. GoldenGate monitors the database for changes, which are routed to the adapter and then to Coherence.
2. Coherence instances are started and the cache warmer prepopulates Coherence with relevant data.
3. Coherence applications then work normally and query the database.
4. Database applications access the database performing changes.
5. GoldenGate informs the adapter of the change, which is then inserted into Coherence.

## Configuring the GoldenGate HotCache Adapter

To configure GoldenGate for the HotCache adapter:

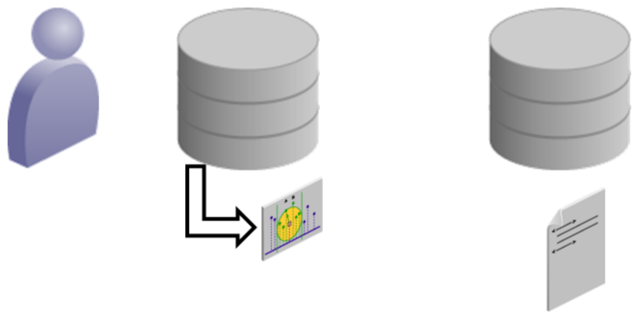

Provision the Database for GoldenGate.
Configure the HotCache Adapter.
Configure the GoldenGate Java Client.
Configure Coherence.
Interact with Coherence and the Database Normally.

12 - 11

## Provisioning the Database

Provisioning a database includes the following steps:

1. Create a user and assign the required privileges.
2. Enable database- and table-level logging.
3. Provision Extract properties.
4. Create and configure Extract.



## Prerequisite: Installing GoldenGate

To enable the GoldenGate HotCache adapter, first install GoldenGate, which involves the following steps:

1. Download and install Oracle GoldenGate.
2. Run `ggcsi` as required.
3. Create and configure `mgr.prm`.
4. Add all JVM libraries to the libraries path.
5. Start the GoldenGate manager.

GoldenGate 12*c* is now available and qualified with the GoldenGate HotCache adapter.

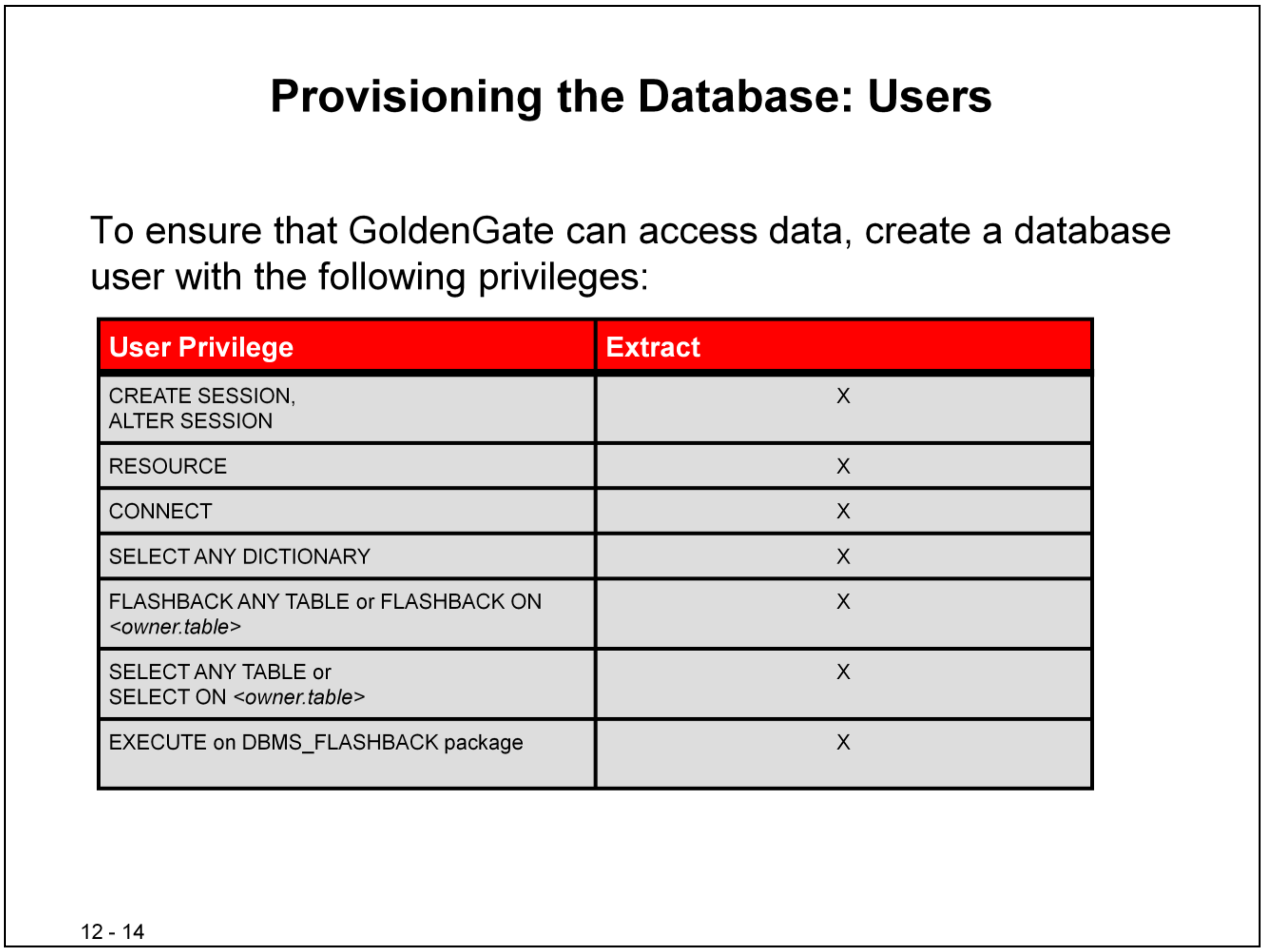
Installing GoldenGate is beyond the scope of this course. See the *Oracle GoldenGate for Oracle Installation and Setup Guide* to install GoldenGate on Oracle databases and *Oracle GoldenGate for Java Administrator's Guide*.

12 - 13

Installing Oracle GoldenGate is beyond the scope of the Coherence Administration class. However, a number of tutorials exist for installing GoldenGate and the Coherence adapters. See www.oracle.com/oll and search for “GoldenGate and Coherence” for more information. For this class, the following were used:

- Oracle GoldenGate on Oracle, Linux-64 (Oracle Software Delivery Cloud ), version 11.2.1.0.2, part number V34339-01 for 64-bit Linux
- Oracle GoldenGate Application Adapters 11.1.1.0.0 for JMS and Flat File Media Pack (Oracle Software Delivery Cloud), version 11.1.1.0.1, part number V22250-01 for 64-bit Linux
- Oracle XE. Note that technically Oracle XE is not a supported environment for Oracle GoldenGate

Note that the GoldenGate Command Interpreter is used to interact with the GoldenGate environment and create, start, and stop the Extract and Java adapter processes that are used to keep the database and Coherence synchronized. Note that `mgr.prm` is used for several reasons, the primary reason being configuring the GoldenGate port.



## Provisioning the Database: Users

To ensure that GoldenGate can access data, create a database user with the following privileges:

| User Privilege | Extract |
|---|---|
| CREATE SESSION,<br>ALTER SESSION | X |
| RESOURCE | X |
| CONNECT | X |
| SELECT ANY DICTIONARY | X |
| FLASHBACK ANY TABLE or FLASHBACK ON *<owner.table>* | X |
| SELECT ANY TABLE or<br>SELECT ON *<owner.table>* | X |
| EXECUTE on DBMS_FLASHBACK package | X |

Although not required, Oracle GoldenGate recommends creating a user specifically for the Oracle GoldenGate application, with all the privileges listed in the slide. To ensure that processing can be monitored accurately, do not permit other users or processes to operate as the Oracle GoldenGate user.

In general, the following permissions are necessary for the Oracle GoldenGate user:

- Permissions to read the data dictionary or catalog tables
- Permissions to select data against the tables

In addition, execute the following command in SQL*Plus as `SYSDBA`:

```
EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE('GGUSER','*',TRUE);
```

where *`GGUSER`* is the database user ID used in the GGSCI `DBLogin` commands.

Note that GoldenGate typically refers to the source and target systems because it is used to transfer or transform data between systems. When used with Coherence, the database is considered the source system.

## Provisioning the Database: Example

Provision the example user, `csdemo`, as required for Oracle GoldenGate.

```
SQL> create USER csdemo IDENTIFIED BY csdemo;
SQL> grant dba TO csdemo;
SQL> grant alter session to csdemo;
SQL> grant create session to csdemo;
SQL> grant flashback any table to csdemo;
SQL> grant select any dictionary to csdemo;
SQL> grant select any table to csdemo;
SQL> grant select any transaction to csdemo;
SQL> grant unlimited tablespace to csdemo;
```

12 - 15

## Provisioning the Database: Monitoring Tables

Minimal logging must be enabled to allow Oracle GoldenGate to properly capture updates.

- Enable logging at the database level.

```
SQL> alter database add supplemental log data;
SQL> alter system switch logfile;
```

- Enable logging at the table level.

```
GGSCI> DBLogin UserID <login>, Password <pswd>
GGSCI> Add TranData <OWNER>.<TABLE1>
GGSCI> Add TranData <OWNER>.<TABLE2>
```

12 - 16

Oracle GoldenGate reads the online logs by default (or the archived logs if it starts falling “behind”).

It is recommended that archive logging be enabled and that you keep the archived logs on the system for the longest time possible (until they are all processed). This prevents the need to resync data if the online logs recycle before all data has been processed.

## GGSCI

The GoldenGate Software Command Interface, or `ggsci`

- Is the command interface between users and Oracle GoldenGate components
- Is used to set up, manage, monitor, and troubleshoot GoldenGate components and configuration
- Can configure extracts, execute shell commands, define trail files, and a variety of other functions

```
$ ./ggsci
GGSCI (SomeHostName) 1> create subdirs
Creating subdirectories . . .
GGSCI (SomeHostName) 2> exit
$
```

12 - 17

The Oracle GoldenGate Software Command Interface or `ggsci` is a tool that is used to control and configure GoldenGate components. The tool can be used for a variety of purposes, including setting up extracts. GoldenGate HotCache requires core GoldenGate configuration in both the database and GoldenGate itself. `ggsci` is used to specify the characteristics of the extract used by HotCache. Subsequent slides show examples of using `ggsci` to configure an extract on a given table.

## Provisioning the Database: Extract from Table – Parameter File Example

Parameter files are used to define and tune extracts.

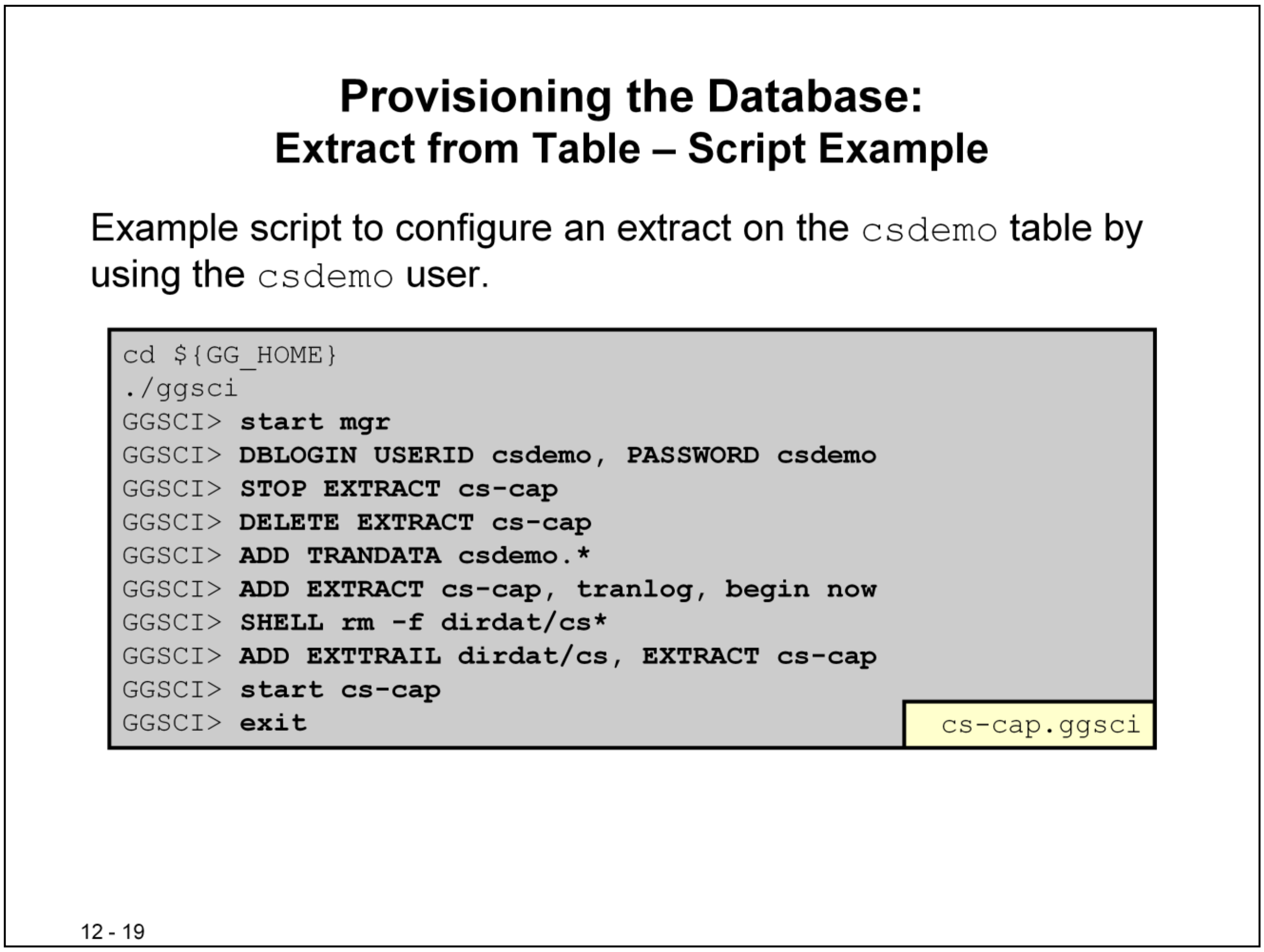
```
EXTRACT cs-cap
USERID csdemo, PASSWORD csdemo
RecoveryOptions OverwriteMode
EXTTRAIL dirdat/cs
BR BROFF
getUpdateBefores
TABLE csdemo.*;
TranLogOptions excludeUser csdemo
```

cs-cap.oby

12 - 18

The current example defines the details of the parameter file used by the an example extract. The parameter file specifically details the following:

- `EXTRACT cs-cap`: This is the cs-cap extract.
- `USERID csdemo`, `PASSWORD csdemo`: Use this user ID and password.
- `RecoveryOptions OverwriteMode`: The existing transaction data is overwritten after the last write-checkpoint position, instead of appending.
- `EXTTRAIL dirdat/cs`: The trail file is identified as **`dirdat/cs`**.
- `BR BROFF`: Disable Bounded Recovery for the run and for recovery.
- `getUpdateBefores`: Specify that Before images of updated columns are included in the records that are processed by Oracle GoldenGate.
- `TABLE csdemo.*`: Monitor all **`csdemo`** tables.
- `TranLogOptions excludeUser csdemo`: Ignore changes made by **`csuser`**.



## Provisioning the Database:
### Extract from Table – Script Example

Example script to configure an extract on the `csdemo` table by using the `csdemo` user.

```
cd ${GG_HOME}
./ggsci
GGSCI> start mgr
GGSCI> DBLOGIN USERID csdemo, PASSWORD csdemo
GGSCI> STOP EXTRACT cs-cap
GGSCI> DELETE EXTRACT cs-cap
GGSCI> ADD TRANDATA csdemo.*
GGSCI> ADD EXTRACT cs-cap, tranlog, begin now
GGSCI> SHELL rm -f dirdat/cs*
GGSCI> ADD EXTTRAIL dirdat/cs, EXTRACT cs-cap
GGSCI> start cs-cap
GGSCI> exit
```

cs-cap.ggsci

12 - 19

Oracle GoldenGate works by using extracts that track specific table changes. The script shown in the slide creates an extract against the `csdemo` table only. Specifically the following script:

- `start mgr`: Start the GoldenGate manager process.
- `DBLOGIN USERID csdemo, PASSWORD csdemo`: Log in to the database.
- `STOP EXTRACT cs-cap`
  `DELETE EXTRACT cs-cap`: Stop and delete any running extract named **`cs-cap`**.
- `ADD TRANDATA csdemo.*`: Monitor any table named **`csdemo*`** for changes.
- `ADD EXTRACT cs-cap`, `tranlog`, `begin now`: Use any parameters in **`cs-cap.prm`** and create an extract.
- `SHELL rm -f dirdat/cs*`: Delete all trail files in the **`dirdat`** directory. This ensures that if the extract is being re-created, there are no old trail files. Note that the **`rm -f`** command is platform-specific.
- `ADD EXTTRAIL dirdat/cs, EXTRACT cs-cap`: Create an extract named **`cs-cap`** by using parameters from the **`dirprm/cs-cap.prm`** file. A trail is added at **`dirdat/cs`** from the extract **`cs-cap`** file.
- `start cs-cap`: Start the **`cs-cap.ggsci`** script.

## HotCache Adapter Configuration: Overview

The Coherence GoldenGate Adapter is configured by using:

- GoldenGate properties file:

```
gg.handlerlist=cache
gg.handler.cache.type=oracle.toplink.goldengate.CoherenceAdapter
    gg.handler.textfile1.type=singlefile
gg.handler.textfile1.format=text
gg.handler.textfile1.mode=op
    . . .
```

- `javawriter.bootoptions`: Direct cluster membership

```
java.class.path elements
-Djava.class.path=.dirprm:./ggjava/ggjava.jar:${COHERENCE_HOME}. . .
/java.x.persistanceXX.jar:${MIDDLEWARE_HOME}. . . /eclipselink.jar . . .
Coherence Properties
-Dtangosol.coherence.localhost=unicast ip address
-Dtangosol.coherence.distributed.localstorage=false
-Dtangosol.coherence.cacheconfig=coherence cache configuration
-Dtangosol.coherence.ttl=0 . . .
Adapter related
-Dtoplink.goldengate.adapter.persistence-unit=xxx
```

12 - 20

The GoldenGate Adapter for Coherence is configured by using two mechanisms. Note that this page is not intended to be an exhaustive list of all the configurations required for the adapter, but rather representative because the following slides contain more concrete examples. See the Oracle Coherence documentation for complete configuration coverage.

**Properties file:** The adapter requires a properties file that specifies a small number of parameters, particularly: that the handler is an adapter, the adapter type, and a small number of other configuration parameters.

**`javawriter.bootproperties`:** The bulk of the adapter is configured by using `javawriter.bootoptions`. The options fall into four specific areas, three of which are shown (logging properties are omitted only for space). They are:

- **`java.class.path` elements:** Elements used to locate the adapter JAR, Coherence libraries, and TopLink (JPA) libraries. In the example shown in the slide, Coherence can be installed stand alone or as part of WebLogic Server. The `MIDDLEWARE_HOME` directory is assumed to be the root of a WebLogic installation directory.
- **Coherence properties:** Traditional Coherence properties, including cache configuration, local storage–disabled, local cache server (TTL=0), and others as required
- **Adapter:** The name of a JPA persistence unit XML file that defines the object-relational mapping used between the database and Coherence
- **Logging:** `-Dlog4j.configuration=`an appropriate logging definition.

## Configuring the HotCache Adapter: Creating a Properties File

```
gg.handlerlist=cache
gg.handler.cache.type=
                  oracle.toplink.goldengate.CoherenceAdapter
goldengate.userexit.nochkpt=true
goldengate.userexit.writers=javawriter
#
# Optionally
#
gg.handler.txtfile1.type=singlefile
gg.handler.txtfile1.format=text
gg.handler.txtfile1.mode=op
gg.handler.txtfile1.file=./dirout/csdemo-output.txt
goldengate.log.logname=./dirout/csdemo-userexit-jcoh
```

cs-cgga.properties

12 - 21

Create a properties file that contains, at a minimum:

- `g.handlerlist=`***`cache`***: Comma-separated list of handlers
- `gg.handler.`***`cache`***`.type=`
  `oracle.toplink.goldengate.CoherenceAdapter`:
  Handler class for each handler. Note that the names match between the handler list and the handler type.
- `goldengate.userexit.nochkpt=true`: User checkpoint disabled on exit
- `goldengate.userexit.writers=javawriter`: The name of the writer; must be **javawriter** to enable calling out to the GoldenGate Java Adapter

Optionally:

```
gg.handler.txtfile1.type=singlefile
gg.handler.txtfile1.format=text
gg.handler.txtfile1.mode=op
gg.handler.txtfile1.file=./dirout/csdemo-output.txt
goldengate.log.logname=./dirout/csdemo-userexit-jcoh
```

## Configuring the HotCache Adapter: Java Writer Boot Options – Classpath

Java Boot Options is broken into four distinct elements:

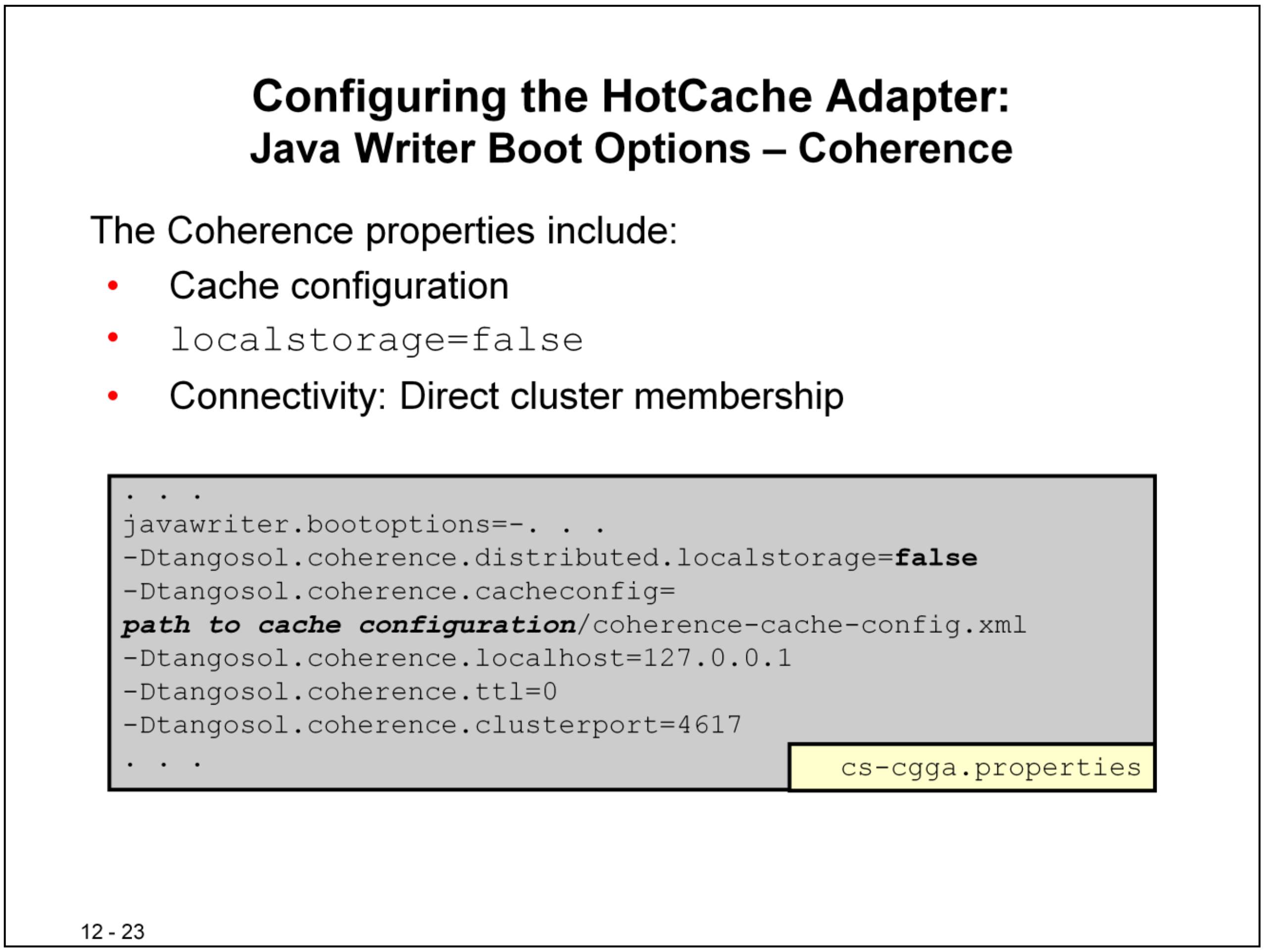
- Classpath-related entries
- HotCache properties
- Coherence properties
- Logging properties

```
javawriter.bootoptions=-Djava.class.path=./dirprm:
./ggjava/ggjava.jar:
${ORACLE_HOME}/ojdbc6.jar:
${COHERENCE_HOME}/coherence.jar:
${MIDDLEWARE_HOME}/oracle_common/modules/javax.persistence_2
.0.0.0_2-0.jar:
${MIDDLEWARE_HOME}/oracle_common/modules/oracle.toplink_12.1
.2/eclipselink.jar:
${MIDDLEWARE_HOME}/oracle_common/modules/oracle.toplink_12.1
.2/toplink-grid.jar:
Application Specific classes
```

`cs-cgga.properties`

12 - 22

The Java writer boot options requires the ODBC JAR, Coherence JAR, javax persistence for JPA, EclipseLink, TopLink, and any application-specific classes as shown in the slide.



## Configuring the HotCache Adapter: Java Writer Boot Options – Coherence

The Coherence properties include:

- Cache configuration
- `localstorage=false`
- Connectivity: Direct cluster membership

```
. . .
javawriter.bootoptions=-. . .
-Dtangosol.coherence.distributed.localstorage=false
-Dtangosol.coherence.cacheconfig=
path to cache configuration/coherence-cache-config.xml
-Dtangosol.coherence.localhost=127.0.0.1
-Dtangosol.coherence.ttl=0
-Dtangosol.coherence.clusterport=4617
. . .
```

cs-cgga.properties

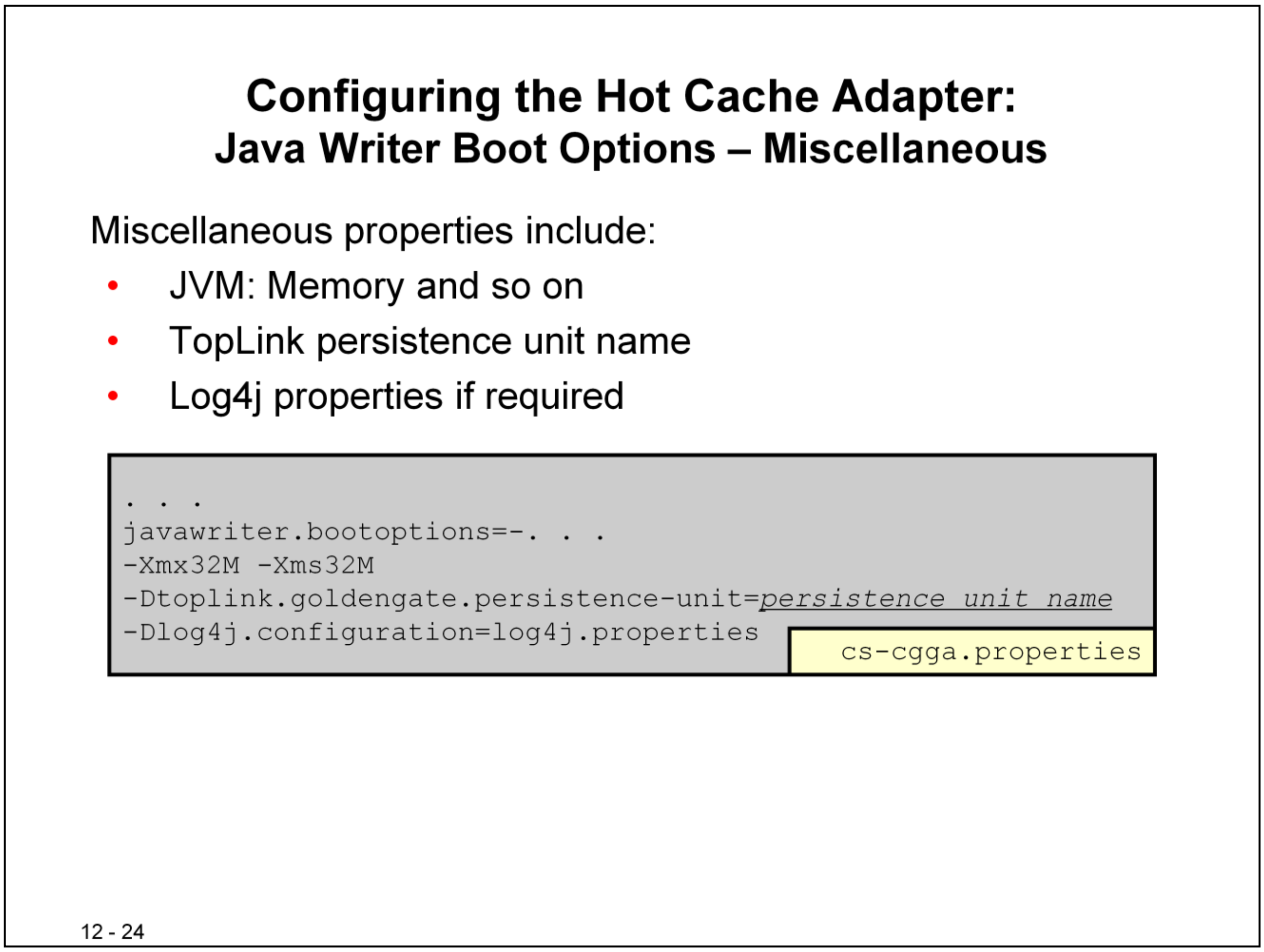
12 - 23

The Coherence properties include:

- `-Dtangosol.coherence.distributed.localstorage=false`: Disable local storage.
- `-Dtangosol.coherence.cacheconfig= path to cache configuration/coherence-cache-config.xml`

Connection-related properties:

`-Dtangosol.coherence.localhost=address of host`, often 127.0.0.1
`-Dtangosol.coherence.ttl=0`: Do not propagate packets.
`-Dtangosol.coherence.clusterport=`*`port`*



## Configuring the Hot Cache Adapter: Java Writer Boot Options – Miscellaneous

Miscellaneous properties include:

- JVM: Memory and so on
- TopLink persistence unit name
- Log4j properties if required

```
. . .
javawriter.bootoptions=-. . .
-Xmx32M -Xms32M
-Dtoplink.goldengate.persistence-unit=persistence unit name
-Dlog4j.configuration=log4j.properties
```

cs-cgga.properties

The remaining properties include:

- `-Xmx32M -Xms32M`: JVM-specific properties such as memory
- `-Dtoplink.goldengate.persistence-unit=employee`: Persistence unit required for TopLink
- `-Dlog4j.configuration=log4j.properties`: Log4J properties file

To allow HotCache to log errors, add the following line to the properties file:

```
-Dlog4j.logger.oracle.toplink.goldengate=DEBUG, stdout, rolling
```

## GoldenGate Java Client

The GoldenGate Java client:

- Provides a way to process GoldenGate data change events
- Is configured as an event handler via a `.prm` file
- Monitors an extract and passes data change events to HotCache

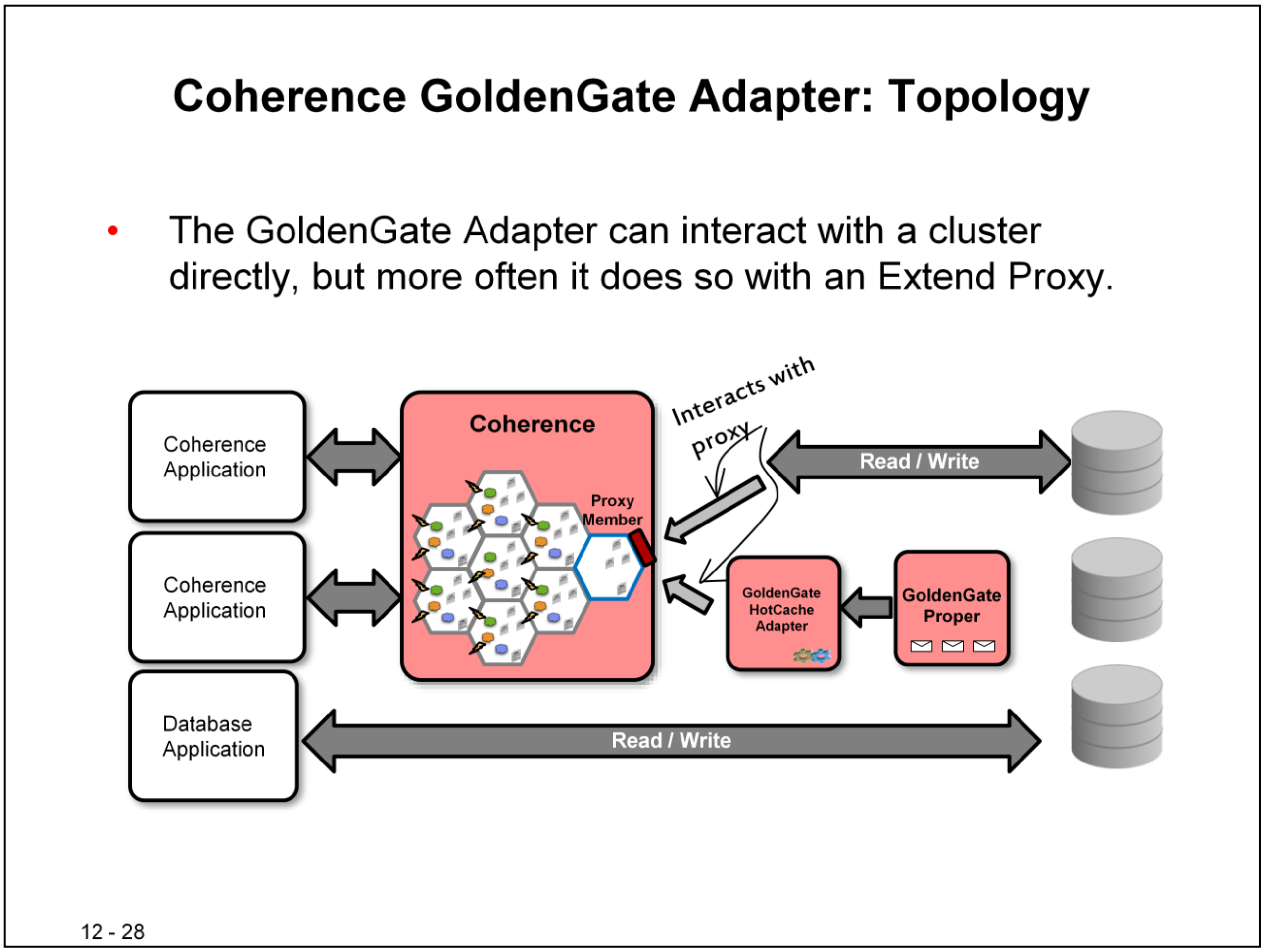
```
Extract cs-cgga
USERID csdemo, PASSWORD csdemo
SetEnv ( GGS_USEREXIT_CONF = "./dirprm/cs-cgga.properties" )
GetEnv (PATH)
CUserExit libggjava_ue.so CUSEREXIT PassThru

IncludeUpdateBefores

. . .
```

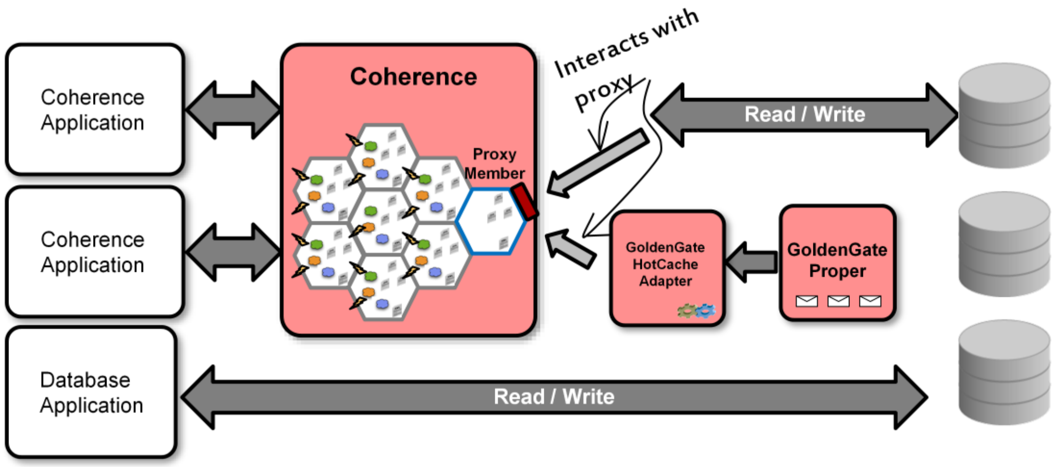
`cs-cgga.prm`

The final element of the HotCache adapter is an event handler that monitors database extracts and passes changes back and forth to the HotCache.



## Coherence GoldenGate Adapter: Topology

- The GoldenGate Adapter can interact with a cluster directly, but more often it does so with an Extend Proxy.



12 - 28

A typical GoldenGate/Coherence topology includes:

- Instances of the GoldenGate Adapter and Java clients that run on a machine that accesses the database. Database clients are not impacted by the running GoldenGate adapter. The adapter has no impact on normal database access. The adapter interacts with Coherence either directly, or via a proxy as shown in the example in the slide.
- The Coherence proxy instance that insulates the Coherence cluster and the GoldenGate database from each other. The proxy instance acts an intermediary, running storage disabled, and sending data requests back and forth between the Coherence cluster and the database.
- The Coherence cluster that acts normally, and is accessed by Coherence applications without concern or knowledge of GoldenGate acting as a synchronizing intermediary.

## Coherence*Extend Connection

```
<cache-config>
. . .
 <caching-schemes>
   <remote-cache-scheme>
     <scheme-name>CustomRemoteCacheScheme</scheme-name>
     <service-name>CustomExtendTcpCacheService</service-name>
     <initiator-config>
       <tcp-initiator>
         <remote-addresses>
           <socket-address>
             <address>localhost</address>
             <port>9099</port>
           </socket-address>
         </remote-addresses>
       </tcp-initiator>
       <outgoing-message-handler>
       . . .
       </outgoing-message-handler>
    </initiator-config>
  </remote-cache-scheme>
...
</cache-config>
```

**Oracle recommends that the connection between HotCache and the Coherence cluster be made with Coherence*Extend.**

The example in the slide illustrates the section of a client cache configuration file that uses Coherence*Extend to connect to the Coherence cluster. In the client cache configuration file, Coherence*Extend is configured in the `<remote-cache-scheme>` section. By default, the connection port for Coherence*Extend is **9099**. Note that this section represents the client connection. The server configuration is shown as an example on the next page.

## Summary

In this lesson, you should have learned how to:

- Explain GoldenGate and HotCache adapter concepts
- Explain JPA concepts
- Configure GoldenGate for HotCache

12 - 31