Advanced Repository Concepts

Presenter's Name
Presenter's Title

Enablement 2.0
DEVELOP · SELL · IMPLEMENT

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.
The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Oracle Training Materials – Usage Agreement

Use of this Site ("Site") or Materials constitutes agreement with the following terms and conditions:

1. Oracle Corporation ("Oracle") is pleased to allow its business partner ("Partner") to download and copy the information, documents, and the online training courses (collectively, "Materials") found on this Site. The use of the Materials is restricted to the non-commercial, internal training of the Partner's employees only. The Materials may not be used for training, promotion, or sales to customers or other partners or third parties.

2. All the Materials are trademarks of Oracle and are proprietary information of Oracle. Partner or other third party at no time has any right to resell, redistribute or create derivative works from the Materials.

3. Oracle disclaims any warranties or representations as to the accuracy or completeness of any Materials. Materials are provided "as is" without warranty of any kind, either express or implied, including without limitation warranties of merchantability, fitness for a particular purpose, and non-infringement.

4. Under no circumstances shall Oracle or the Oracle Authorized Delivery Partner be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this Site of Materials. As a condition of use of the Materials, Partner agrees to indemnify Oracle from and against any and all actions, claims, losses, damages, liabilities and expenses (including reasonable attorneys' fees) arising out of Partner's use of the Materials.

5. Reference materials including but not limited to those identified in the Boot Camp manifest can not be redistributed in any format without Oracle written consent.

Agenda

- SQL Repository Cache
- Versioned Repository
- Types of ATG Repositories

Learning Objectives

At the end of this lesson, you should be able to:

- Apply caching strategies to optimize access to the database
- Configure Cache mode on repository items and monitor cache usage
- Preload and flush repository cache
- Create and register a versioned repository
- Learn about the various types of ATG repositories

Section 1:
SQL Repository Cache

SQL Repository Cache Overview

- Caching is a process of storing data read from DB into memory for later use.
- Caching minimizes access to the database and ensures data integrity.
- It improves performance.
- ATG SQL repository includes two types of caches:
  - Item cache,
  - Query cache.
- An intelligent caching strategy is central to:
  - Efficient database access,
  - Minimize access to database,
  - Ensure data integrity.

Item Cache

- Item cache holds the values of repository items in memory, indexed by repository IDs.
- It can be explicitly enabled for each item descriptor.
- Item caching occurs within the scope of each transaction.
- An item cache entry is invalidated when that item is updated.
- The scope of the entry's invalidation depends on its caching mode.
  - Simple caching only invalidates local cache.

Query Cache

- Query cache holds the repository IDs of items that match given queries.
- Subsequent iterations of this query use the query cache's result set and cached items. Any items that are missing from the item cache are fetched again from the database.
- Query caching is turned off by default.
- A query cache entry can be invalidated for two reasons:
  - A cached item property that was specified in the original query is modified.
  - Items of a queried item type are added to or removed from the repository.

Cache Modes

- The following are the cache modes supported in ATG:
  - Disabled,
  - Simple caching (default),
  - Distributed:
    - Distributed TCP caching,
    - Distributed JMS caching,
    - Distributed hybrid caching,
  - Locked caching.

Cache Mode : Disabled

- Caching can be disabled for specific items or properties:
  - To set an item type's caching mode, set its <item-descriptor> tag's cache-mode attribute to disabled.
  - To set it for the property, set the <property> tag's cache-mode attribute to disabled. Each property's definition overrides the caching mode set for its item descriptor.
- Used when data in underlying data source changes too often or for security reasons:

```
<item-descriptor name="user" cache-mode="simple">
  <table name="dps_user">
    <property name="password" cache-mode="disabled">
    ...
  </table>
  ...
```

Cache Mode : Simple

- In simple cache mode, each server maintains its own cache in memory.
- A server obtains changes to an item’s persistent state only after the cached entry for that item is invalidated.
- This mode is suitable for read-only repositories.
- It has the highest performance of all cache modes:

```
<item-descriptor name="user" cache-mode="simple">
  <table name="dps_user">
    <property name="password" cache-mode="disabled">
    ...
  </table>
  ...
```

Cache Mode : Distributed TCP

- Distributed TCP cache is synchronized between multiple ATG instances.
- A cache invalidation event is broadcast from that ATG instance to all other ATG instances that use distributed TCP caching.
- It is suitable when:
  - Changes to an item on one instance need to be propagated to others.
  - Items are subject to frequent reads.
  - Items are rarely changed, added, or deleted.

Cache Mode : Distributed JMS

- Distributed JMS cache sends invalidation events using JMS.
- It guarantees that all servers receive cache invalidation events.
- It is used for items that are infrequently updated.
- Performance for distributed JMS caching is much worse than using distributed TCP caching.

Cache Mode : Distributed Hybrid

- Distributed hybrid cache uses intelligent cache invalidation:
  - Cache invalidation events are sent only to servers where the items are cached.
- Distributed hybrid caching is suitable for sites with the following requirements:
  - Real-time access to item changes,
  - Large number of items to monitor across many clients.

Cache Mode : Locked

- A multi-server application might require locked caching where only one ATG instance at a time has write access to the cached data of a given item type.
- Locked caching has the following prerequisites:
  - Disable query cache,
  - Use ClientLockManager,
  - At least one ClientLockManager on each ATG instance configured to use a ServerLockManager,
  - Configure a ServerLockManager.

Cache Configuration : Item Cache

- Configure item caches with the following <item-descriptor> attributes:
  - item-cache-size,
  - item-expire-timeout,
  - item-cache-timeout.

```
<item-descriptor name="order" cache-mode="simple"
   item-cache-timeout="180000"
   item-cache-size="1000"
   item-expire-timeout="180000">
   ...
</item-descriptor>
```

Cache Configuration : Query Cache

- Configure query caches with the following <item-descriptor> attributes:
  - query-cache-size,
  - query-expire-timeout.

```
<item-descriptor name=“user"
   query-cache-size=“5000”
   query-expire-timeout=“180000” >
   ...
</item-descriptor>
```

Query Cache Tuning

- A query cache size should anticipate the number of queries that are typically executed against that repository.
- It is generally safe to set the size of the query cache to 1000 or higher. Query caches only contain the query parameters and string IDs of the result set items, so large query cache sizes can usually be handled comfortably without running out of memory.

Limit the Lifetime of Cached Items

- An item descriptor can limit the lifetime of cached items in two ways.
  - **Force refreshes of items** in the item and query caches: An item descriptor’s item-expire-timeout and query-expire-timeout attributes specify how long items can stay in the item and query caches.

```
<item-descriptor name=“user"
   query-expire-timeout=“180000”
   item-expire-timeout=“180000”>
   ...
</item-descriptor>
```

  - **Refresh unused item** cache entries: Use item-cache-timeout attribute.

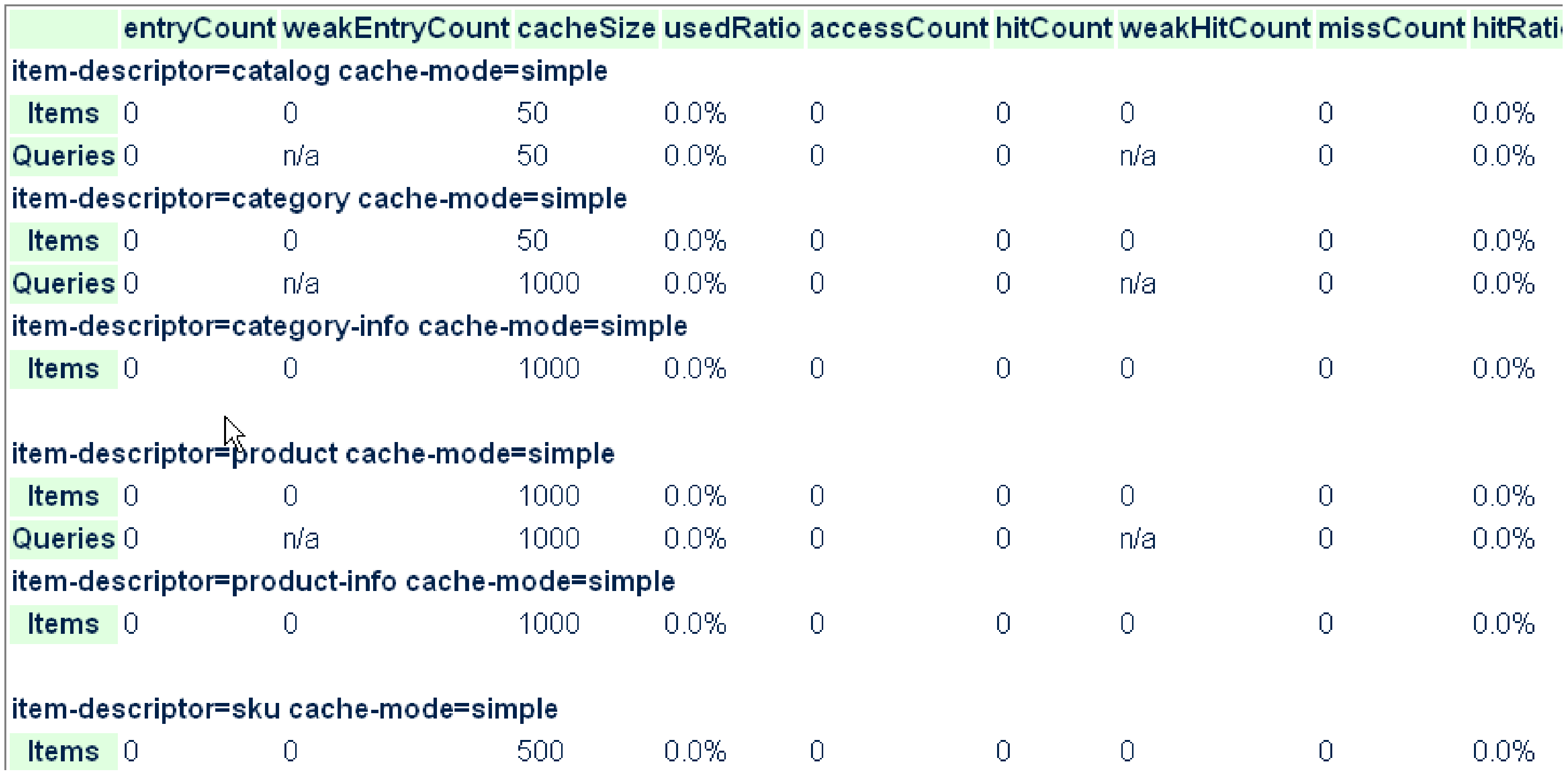
```
<item-descriptor name=“user"
   item-cache-timeout=“180000”>
   ...
</item-descriptor>
```

Monitoring Cache Usage

- You can view details on usage of repository caches with the Administrative Interface Component Browser.
- The following table lists some important attributes:

Property	Description
accessCount	Total tries, successful and unsuccessful, to retrieve items or query results since cache startup.
hitCount	Total number of successful access tries since cache startup.
hitRatio	The hitCount percentage of accessCount. For example, if accessCount is 100 and hitCount is 75, hitRatio is 75%.

Example : Monitoring Cache Usage



	entryCount	weakEntryCount	cacheSize	usedRatio	accessCount	hitCount	weakHitCount	missCount	hitRati
item-descriptor=catalog cache-mode=simple									
Items	0	0	50	0.0%	0	0	0	0	0.0%
Queries	0	n/a	50	0.0%	0	0	n/a	0	0.0%
item-descriptor=category cache-mode=simple									
Items	0	0	50	0.0%	0	0	0	0	0.0%
Queries	0	n/a	1000	0.0%	0	0	n/a	0	0.0%
item-descriptor=category-info cache-mode=simple									
Items	0	0	1000	0.0%	0	0	0	0	0.0%
item-descriptor=product cache-mode=simple									
Items	0	0	1000	0.0%	0	0	0	0	0.0%
Queries	0	n/a	1000	0.0%	0	0	n/a	0	0.0%
item-descriptor=product-info cache-mode=simple									
Items	0	0	1000	0.0%	0	0	0	0	0.0%
item-descriptor=sku cache-mode=simple									
Items	0	0	500	0.0%	0	0	0	0	0.0%

Caching by Repository IDs

- It is inefficient to retrieve and cache the child items each time the parent item is cached, so you can specify to cache only the repository IDs of the child items.
- This can be done by setting the property’s cacheReferencesById to true:

```
<property name="childProducts" ...>
    ...
   <attribute name="cacheReferencesById" value="true"/>
</property>
```

Preloading Caches

- You can achieve better performance in an application that uses SQL repositories by preloading caches.
- A good practice is to put cache-loading tags in a separate XML file with the same name as the repository definition file.
- You can specify to load certain items into repository caches on application startup in several ways:
  - Load specific repository items,
  - Load queried items,
  - Load from a dump log.

Cache Flushing

- The ATG distribution provides class methods that you can use to explicitly flush item and query caches at several levels:
  - Flushing all repository caches,
  - Flushing item caches,
  - Flushing query caches.
- The following table summarizes these methods:

Class	Methods
atg.repository.RepositoryImpl	invalidateCaches()

atg.repository.ItemDescriptorImpl	removeItemFromCache()
	invalidateItemCache()
	invalidateCaches()

atg.repository.RepositoryViewImpl	invalidateQueryCache()

Section 1 ✓
Check Your Understanding

When is the item cache entry invalidated?

Answer:

When an item is updated.

Section 1 ✓
Check Your Understanding

At what levels can you flush caches explicitly in ATG?

Answer:

Item level, query level, and all repository caches.

Section 1 ✓
Check Your Understanding

What is item-cache-timeout used for? How is it different from item-expire-timeout?

Answer:

Item-cache-timeout is the timeout for unused item. Item-expire-timeout is how long items can stay in the item cache before they are refreshed.

Section 1 ✓
Check Your Understanding

When only one ATG instance at a time has to have the write access to the cached data, which cache mode would you use?

Answer:

Locked caching.

Section 1 ✓
Check Your Understanding

What is the difference between distributed TCP and distributed JMS caching?

Answer:

Distributed TCP sends invalidation events through TCP protocol. Distributed JMS uses JMS. Distributed JMS is much lower performing than distributed TCP, but guarantees message delivery to all servers.

Section 1 ✓
Check Your Understanding

Which cache mode has the highest performance of all cache modes?

Answer:

Simple cache mode has the highest performance.

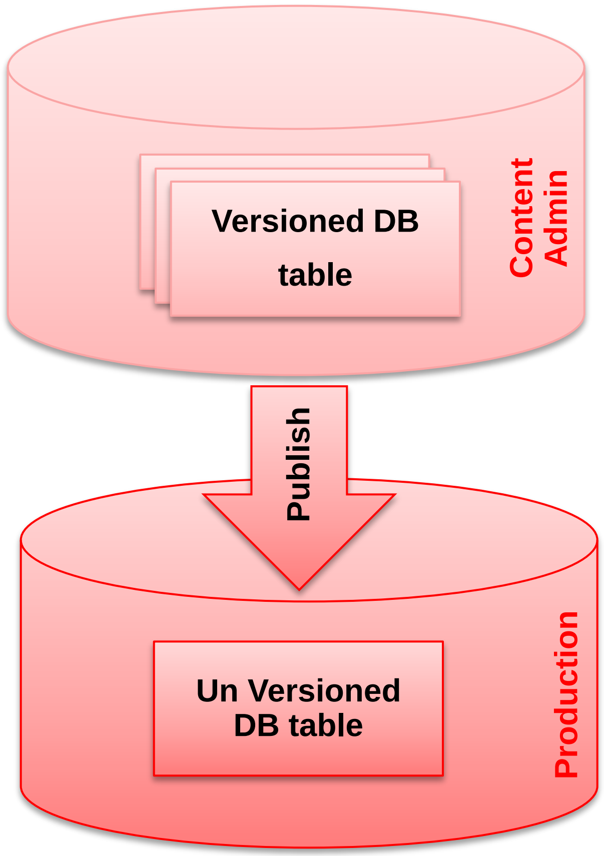
Summary

- Caching is a process of storing data read from DB into memory for later use.
- ATG SQL repository includes two types of caches: item and query cache.
- Item cache holds the values of repository items in memory, indexed by repository IDs.
- Query cache holds the repository IDs of items that match given queries.
- ATG supports the following cache modes: disabled, simple, distributed, and locked.

Section 2:
Versioned Repository

Versioned Repositories Overview

- Versioned repository is a way to manage assets.
- You can track each change to asset versions.
- Provides an option to revert to an earlier version.
- ATG Content Administration uses versioned repositories to manage repository content assets.



Creating a Versioned Repository : Overview

- Configuring the asset management server to support your application’s repository assets involves these general steps:
  - Create and configure versioned repositories.
  - Create and install the versioned database schema.

Create and Configure Versioned Repositories

- Verify the versioned repository definition.
- Modify the repository configuration.
- Modify the repository definition, if necessary.
- Set the behavior of versioning caches. (optional)
- Modify the secured repository definition.

Create and Install the Versioned Database Schema

- Create the versioned schema.
- Configure a versioned repository to use shared tables.
- Install the versioned database schema.

Steps to Create Versioned Schema (1)

- Copy original DDL from un-versioned module.
- Add the following columns to every primary table:
  - `asset_version` INT NOT NULL
  - `workspace_id` VARCHAR(40) NOT NULL
  - `branch_id` VARCHAR(40) NOT NULL
  - `is_head` NUMERIC(1) NOT NULL
  - `version_deleted` NUMERIC(1) NOT NULL
  - `version_editable` NUMERIC(1) NOT NULL
  - `pred_version` INT NULL
  - `checkin_date` TIMESTAMP NULL
- Add this column to every auxiliary table:
  - `asset_version` INT NOT NULL

Steps to Create Versioned Schema (2)

- Change all primary keys to composite primary keys, composed of the original primary key column(s) and the asset_version column.
- Create an index for the workspace_id and checkin_date columns that are added to each primary table.
- Remove from all tables:
  - All foreign key references.
  - All unique constraints on columns. Also remove unique attributes from all <property> tags. For more information, see the discussion on unique properties in the ATG Repository Guide.
  - All unique indexes on columns.

Register the Versioned Repositories

- All versioned repositories must be registered with ATG Content Administration as follows:
  - In the asset management server's localconfig layer, create this properties file : **`/atg/registry/ContentRepositories.properties`**
  - Set its $class property as follows : $class=atg.repository.nucleus.RepositoryRegistryService
  - Add the versioned repositories to the list property initialRepositories, in this format : initialRepositories+=\versioned-repository [,versioned-repository]...

```
initialRepositories+=\
   /atg/myApp/MyRepositoryVer1,\
   /atg/myApp/MyRepositoryVer2
```

Section 2 ✓
Check Your Understanding

What is the property of ContentRepositories component that you need to add the versioned repository to?

Answer:

initialRepositories property.

Section 2 ✓
Check Your Understanding

What change needs to be performed to primary keys in tables to enable versioned schema?

Answer:

Change all primary keys to composite primary keys, composed of the original primary key column(s) and the asset_version column.

Section 2 ✔
Check Your Understanding

Name the column that needs to be added to all tables, primary and auxiliary.

Answer:

asset_version column.

Section 2 ✓
Check Your Understanding

If you need the ability to revert to an earlier version, which repository would you use?

Answer:

Versioned repository.

Section 2 ✓
Check Your Understanding

What are the steps for configuring the asset management server to support your application's repositories?

Answer:

Create and configure versioned repositories, create and install the versioned database schema.

Summary

- Versioned repository is a way to manage assets that tracks each change to asset versions.
- Create and configure the versioned repositories and install the versioned database schema to configure a versioned repository.
- A list of 8 properties need to be added to each primary table. You need to add 1 property to auxiliary table.
- All versioned repositories must be registered with ATG Content Administration.

Section 3:
Types of ATG Repositories

ATG Repositories

- ATG Data Anywhere Architecture provides a unified view of content and data across the business.
- The core of the ATG Data Anywhere Architecture is the repository API.
- Through the API, you can employ a single approach to accessing disparate data types.
- The typical repository types in ATG are:
  - SQL repository,
  - SQL content repository,
  - Composite repository,
  - LDAP repository,
  - Integration repository.

SQL Repository

- ATG SQL repository is based on SQL database.
- A SQL database provides fast, scalable storage and retrieval of persistent information.
- SQL repository works with a SQL database to store objects and make those objects visible inside an ATG application as Dynamic Beans.
- SQL repositories are the most common type of repositories in ATG.
- SQL repositories are used in ATG for:
  - User profiles,
  - Website content,
  - Catalog data, etc.

SQL Content Repositories

- SQL content repository is comprised of repository items that correspond to documents maintained in a hierarchical namespace.
- It servers as a source of content to display to a user.
- The essential feature of a content repository is that it represents a hierarchical structure of folders and repository items, like a directory structure.
- A repository can contain content repository items and non-content repository items.
- Example of SQL content repositories:
  - Media elements in Commerce.

LDAP Repositories

- LDAP (Lightweight Directory Access Protocol) directories are widely used to store personnel information and other kinds of data across multiple applications.
- The ATG LDAP repository is an implementation of the repository API that enables you to store and access profile data in an LDAP directory.
- The LDAP repository accesses data in the underlying LDAP directory using JNDI (Java Naming and Directory Interface).

Composite Repositories

- The composite repository lets you use more than one data store as the source of a single repository.
- It consolidates all data sources in a single data model, making the data model flexible enough to support the addition of new sources.
- Composite repositories allow all the properties in each composite repository item to be queryable.
- Its primary purpose is to make any number of repositories appear in an ATG application as a single repository.
- Example:
  - You can use composite repositories to manage user profiles for which some of the properties are from a SQL database and some of the properties are from a LDAP repository.

ATG Integration Repository

- ATG integration framework provides the integration repository, which has the ability to represent data on external systems as ATG repository items.
- You can execute queries from ATG against remote systems. The results are returned as repository items.
- You can create queries using RQL or ATG query builder techniques.
- You can submit changes to the repository items and ATG will translate the changes into the format required by the remote system.

Section 3 ✓
Check Your Understanding

Which repository provides you the means to update data on an external system using RQL queries?

Answer:

Integration repository.

Section 3 ✓
Check Your Understanding

What is the primary purpose of the composite repository?

Answer:

Composite repository's primary purpose is to make any number of repositories appear in an ATG application as a single repository.

Section 3 ✓
Check Your Understanding

How does the LDAP repository access data in the underlying LDAP repository?

Answer:

Using JNDI.

Section 3 ✔
Check Your Understanding

Can a repository contain both content repository items and non-content repository items?

Answer:

Yes, a repository can contain content repository items and non-content repository items.

Section 3 ✓
Check Your Understanding

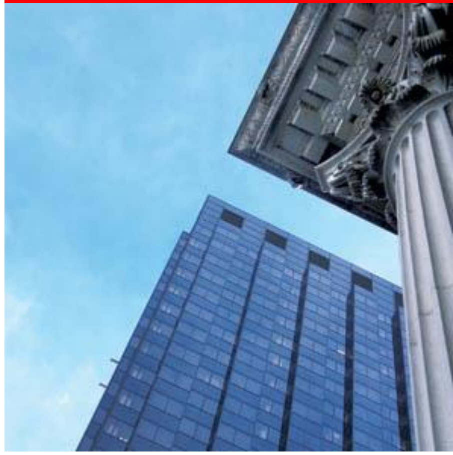
What is the SQL repository used for in ATG?

Answer:

For user profiles, website content, product catalog, etc.

Summary

- ATG Data Anywhere Architecture provides a unified view of content and data across the business.
- **SQL repositories** are the most common type of repositories in ATG and use SQL database as the backend.
- The essential feature of a **content repository** is that it represents a hierarchical structure of folders and repository items, like a directory structure.
- The ATG **LDAP repository** is an implementation of the repository API that enables you to store and access profile data in an LDAP directory.
- The **composite repository** lets you use more than one data store as the source of a single repository.
- ATG **integration repository** has the ability to represent data on external systems as ATG repository items.



Q&A

ORACLE IS THE INFORMATION COMPANY

ORACLE®