39

Creating and Managing Read-Only Materialized Views

You can create and manage read-only materialized views and refresh groups. You can also refresh materialized views.

- Creating Read-Only Materialized Views
  Create a read-only materialized view to replicate a master table's data in a materialized view database.
- Creating Refresh Groups
  Add materialized views to a refresh group to ensure transactional consistency between the related materialized views in the refresh group.
- Refreshing Materialized Views
  Refreshing a materialized view synchronizes the data in the materialized view's master(s) and the data in the materialized view.
- Determining the Fast Refresh Capabilities of a Materialized View
  You can determine whether a materialized view is fast refreshable by attempting to create the materialized view with the `REFRESH FAST` clause or by using the `DBMS_MVIEW.EXPLAIN_MVIEW` procedure.
- Adding a New Materialized View Database
  After you have created a materialized view environment with one or more materialized view databases, you might need to add new materialized view databases.
- Monitoring Materialized View Logs
  You can run queries to display information about the materialized view logs at a master database.
- Monitoring Materialized Views
  You can run queries to display information about the materialized views and refresh groups.

39.1 Creating Read-Only Materialized Views

Create a read-only materialized view to replicate a master table's data in a materialized view database.

Before creating a materialized view to replicate data between a master database and a materialized view database, the database links you plan to use must be available.

1. Connect to the database as a user with the required privileges to create a materialized view.
2. Run the `CREATE MATERIALIZED VIEW` statement.

Example 39-1 Creating a Primary Key Materialized View

```
CREATE MATERIALIZED VIEW hr.employees_mv1 WITH PRIMARY KEY
  AS SELECT * FROM hr.employees@orc1.example.com;
```

Example 39-2 Creating a ROWID Materialized View

```
CREATE MATERIALIZED VIEW oe.orders REFRESH WITH ROWID AS
  SELECT * FROM oe.orders@orc1.example.com;
```

Example 39-3 Creating an Object Materialized View

After the required types are created at the materialized view database, you can create an object materialized view by specifying the `OF` *type* clause.

For example, suppose the following SQL statements create the `oe.categories_tab` object table at the `orc1.example.com` master database:

```
CREATE TYPE oe.category_typ AS OBJECT
   (category_name           VARCHAR2(50),
    category_description    VARCHAR2(1000),
    category_id             NUMBER(2));
/

CREATE TABLE oe.categories_tab OF oe.category_typ
    (category_id    PRIMARY KEY);
```

To create materialized views that can be fast refreshed based on the `oe.categories_tab` master table, create a materialized view log for this table:

```
CREATE MATERIALIZED VIEW LOG ON oe.categories_tab WITH OBJECT ID;
```

The `WITH OBJECT ID` clause is required when you create a materialized view log on an object table.

After you create the `oe.category_typ` type at the materialized view database with the same object identifier as the same type at the master database, you can create an object materialized view based on the `oe.categories_tab` object table using the `OF` *type* clause, as in the following SQL statement:

```
CREATE MATERIALIZED VIEW oe.categories_objmv OF oe.category_typ
   REFRESH FAST
   AS SELECT * FROM oe.categories_tab@orc1.example.com;
```

Here, *type* is `oe.category_typ`.

> **Note:**
>
> The types must be the same at the materialized view database and master database. See "Type Agreement at Replication Databases" for more information.

See Also:

- Read-Only Materialized View Concepts for several examples that create materialized views
- "Required Privileges for Materialized View Operations" for information about the privileges required to create materialized views
- "Required Database Links for Materialized Views"
- "Materialized Views Based on Object Tables"
- *Oracle Database SQL Language Reference*

39.2 Creating Refresh Groups

Add materialized views to a refresh group to ensure transactional consistency between the related materialized views in the refresh group.

When a refresh group is refreshed, all materialized views that are added to a particular refresh group are refreshed at the same time.

1. Connect to the materialized view database as an administrative user with the required privileges to create a refresh group and add materialized views to it.
2. Run the `DBMS_REFRESH.MAKE` procedure to create the refresh group.
3. Run the `DBMS_REFRESH.ADD` procedure one or more times to add materialized views to the refresh group.

Example 39-4 Creating a Refresh Group

This example creates a refresh group and adds two materialized views to it.

```
BEGIN
   DBMS_REFRESH.MAKE (
      name => 'mviewadmin.hr_refg',
      list => '',
      next_date => SYSDATE,
      interval => 'SYSDATE + 1/24',
      implicit_destroy => FALSE,
      rollback_seg => '',
      push_deferred_rpc => TRUE,
      refresh_after_errors => FALSE);
END;
/

BEGIN
    DBMS_REFRESH.ADD (
      name => 'mviewadmin.hr_refg',
      list => 'hr.countries_mv1',
      lax => TRUE);
END;
/

BEGIN
    DBMS_REFRESH.ADD (
      name => 'mviewadmin.hr_refg',
      list => 'hr.departments_mv1',
      lax => TRUE);
```

```
END;
/
```

> **See Also:**
>
> "Refresh Groups"

39.3 Refreshing Materialized Views

Refreshing a materialized view synchronizes the data in the materialized view's master(s) and the data in the materialized view.

You can either refresh all of the materialized views in a refresh group at once, or you can refresh materialized views individually. If you have applications that depend on multiple materialized views at a materialized view database, then Oracle recommends using refresh groups so that the data is transactionally consistent in all of the materialized views used by the application.

1. Connect to the materialized view database as a user with the required privileges to refresh a refresh group or an individual materialized view.
2. Do one of the following:
   - Run the `DBMS_REFRESH.REFRESH` procedure to refresh a refresh group.
   - Run the `DBMS_MVIEW.REFRESH` procedure to refresh an individual materialized view.

Example 39-5 Refreshing a Refresh Group

The following example refreshes the `hr_refg` refresh group:

```
EXECUTE DBMS_REFRESH.REFRESH ('hr_refg');
```

Example 39-6 Refreshing an Individual Materialized View

The following example refreshes the `hr.departments_mv` materialized view:

```
BEGIN
   DBMS_MVIEW.REFRESH (
     list   =>  'hr.departments_mv',
     method =>  '?');
END;
/
```

> **Note:**
>
> Do not use the `DBMS_MVIEW.REFRESH_ALL_MVIEWS` or the `DBMS_MVIEW.REFRESH_DEPENDENT` procedure to refresh materialized views. Instead, use the `DBMS_REFRESH.REFRESH` or the `DBMS_MVIEW.REFRESH` procedure to refresh materialized views in a replication environment.

> **See Also:**
>
> - "Required Privileges for Materialized View Operations" for information about the privileges required to create materialized views
> - *Oracle Database PL/SQL Packages and Types Reference* for more information about the `DBMS_MVIEW` package

39.4 Determining the Fast Refresh Capabilities of a Materialized View

You can determine whether a materialized view is fast refreshable by attempting to create the materialized view with the `REFRESH FAST` clause or by using the `DBMS_MVIEW.EXPLAIN_MVIEW` procedure.

A fast refresh uses materialized view logs to update only the rows that have changed since the last refresh. To determine whether a materialized view is fast refreshable, create the materialized view with the `REFRESH FAST` clause. Oracle Database returns errors if the materialized view violates any restrictions for subquery materialized views. If you specify force refresh, then you might not receive any errors because, when a force refresh is requested, Oracle Database automatically performs a complete refresh if it cannot perform a fast refresh.

You can also use the `EXPLAIN_MVIEW` procedure in the `DBMS_MVIEW` package to determine the following information about an existing materialized view or a proposed materialized view that does not yet exist:

- The capabilities of a materialized view
- Whether each capability is possible
- If a capability is not possible, then why it is not possible

This information can be stored in a varray or in the `MV_CAPABILITIES_TABLE`. To store the information in the table, before you run the `EXPLAIN_MVIEW` procedure, you must build this table by running the `utlxmv.sql` script in the *Oracle_home*`/rdbms/admin` directory.

To determine the fast refresh capabilities of a materialized view:

1. Connect to the materialized view database as an administrative user.
2. Do one of the following:
   - Create the materialized view with the `REFRESH FAST` clause.
   - Run the `DBMS_MVIEW.EXPLAIN_MVIEW` procedure.

Example 39-7 Creating a Materialized View with the FAST REFRESH Clause

```
CREATE MATERIALIZED VIEW oe.orders REFRESH FAST AS
 SELECT * FROM oe.orders@orc1.example.com o
 WHERE EXISTS
   (SELECT * FROM oe.customers@orc1.example.com c
    WHERE o.customer_id = c.customer_id AND c.credit_limit > 10000);
```

Example 39-8 Determining the Refresh Capabilities of an Existing Materialized View

For example, to determine the capabilities of the `oe.orders` materialized view, enter:

```
EXECUTE DBMS_MVIEW.EXPLAIN_MVIEW ('oe.orders');
```

Example 39-9 Determining the Refresh Capabilities of a Materialized View That Does Not Yet Exist

Or, if the materialized view does not yet exist, then you can supply the query that you want to use to create it:

```
BEGIN
  DBMS_MVIEW.EXPLAIN_MVIEW ('SELECT * FROM oe.orders@orc1.example.com o
    WHERE EXISTS (SELECT * FROM oe.customers@orc1.example.com c
    WHERE o.customer_id = c.customer_id AND c.credit_limit > 500)');
END;
/
```

Query the `MV_CAPABILITIES_TABLE` to see the results.

Query the `MV_CAPABILITIES_TABLE` to see the results.

> **Note:**
>
> The `MV_CAPABILITIES_TABLE` does not show materialized view refresh capabilities that depend on prebuilt container tables. For example, complete refresh is required after a partition maintenance operation on a prebuilt container table, but the `MV_CAPABILITIES_TABLE` does not show this limitation.

> **See Also:**
>
> - "Restrictions for Materialized Views with Subqueries"
> - "Materialized View Log"
> - *Oracle Database Data Warehousing Guide* for more information about the `EXPLAIN_MVIEW` procedure

39.5 Adding a New Materialized View Database

After you have created a materialized view environment with one or more materialized view databases, you might need to add new materialized view databases.

You might encounter problems when you try to perform a fast refresh on the materialized views you create at a new materialized view database if both of the following conditions are true:

- Materialized views at the new materialized view database and existing materialized views at other materialized view databases are based on the same master table.
- Existing materialized views can be refreshed while you create the new materialized views at the new materialized view database.

The problem arises when the materialized view logs for the master tables are purged before a new materialized view can perform its first fast refresh. If this happens and you try to perform a fast refresh on the materialized views at the new materialized view database, then you might encounter the following errors:

```
ORA-12004 REFRESH FAST cannot be used for materialized view materialized_view_name
ORA-12034 materialized view log on materialized_view_name younger than last refresh
```

If you receive these errors, then the only solution is to perform a complete refresh of the new materialized view. To avoid this problem, create a dummy materialized view at the new materialized view database before you create your production materialized views. The dummy materialized view ensures that the materialized view log will not be purged while your production materialized views are being created.

If you choose to create a dummy materialized view at the materialized view database, complete the following steps:

1. Create a dummy materialized view called `dummy_mview` based on the master table. For example, to create a dummy materialized view based on a master table named `sales`, issue the following statement at the new materialized view database:

   ```
   CREATE MATERIALIZED VIEW dummy_mview REFRESH FAST AS
     SELECT * FROM pr.sales@orc1.example.com WHERE 1=0;
   ```

2. Create your production materialized views at the new materialized view database.
3. Perform fast refresh of your production materialized views at the new materialized view database.
4. Drop the dummy materialized view.

39.6 Monitoring Materialized View Logs

You can run queries to display information about the materialized view logs at a master database.

- Listing Information About the Materialized View Logs at a Master Database
  A materialized view log enables you to perform a fast refresh on materialized views based on a master. A master can be a master table or a master materialized view.
- Listing the Materialized Views that Use a Materialized View Log
  More than one materialized view can use a materialized view log.

39.6.1 Listing Information About the Materialized View Logs at a Master Database

A materialized view log enables you to perform a fast refresh on materialized views based on a master. A master can be a master table or a master materialized view.

If you have materialized view logs based at a master, then you can use the query in this section to list the following information about them:

- The name of each log table that stores the materialized view log data
- The owner of each materialized view log
- The master on which each materialized view log is based
- Whether a materialized view log is a row id materialized view log
- Whether a materialized view log is a primary key materialized view log
- Whether the materialized view log is an object id materialized view log
- Whether a materialized view log has filter columns

To view this information, complete the following steps:

1. Connect to the master database as an administrative user.
2. Run the following query:

```
COLUMN LOG_TABLE HEADING 'Log Table' FORMAT A20
COLUMN LOG_OWNER HEADING 'Log|Owner' FORMAT A5
COLUMN MASTER HEADING 'Master' FORMAT A15
COLUMN ROWIDS HEADING 'Row|ID?' FORMAT A3
COLUMN PRIMARY_KEY HEADING 'Primary|Key?' FORMAT A7
COLUMN OBJECT_ID HEADING 'Object|ID?' FORMAT A6
COLUMN FILTER_COLUMNS HEADING 'Filter|Columns?' FORMAT A8

SELECT DISTINCT LOG_TABLE,
       LOG_OWNER,
       MASTER,
       ROWIDS,
       PRIMARY_KEY,
       OBJECT_ID,
       FILTER_COLUMNS
    FROM DBA_MVIEW_LOGS
    ORDER BY 1;
```

Your output looks similar to the following:

```
                    Log                   Row Primary Object Filter
Log Table           Owner Master          ID? Key?    ID?    Columns?
------------------- ----- --------------- --- ------- ------ --------
MLOG$_COUNTRIES     HR    COUNTRIES       NO  YES     NO     NO
MLOG$_DEPARTMENTS   HR    DEPARTMENTS     NO  YES     NO     NO
MLOG$_EMPLOYEES     HR    EMPLOYEES       NO  YES     NO     NO
MLOG$_JOBS          HR    JOBS            NO  YES     NO     NO
MLOG$_JOB_HISTORY   HR    JOB_HISTORY     NO  YES     NO     NO
MLOG$_LOCATIONS     HR    LOCATIONS       NO  YES     NO     NO
MLOG$_REGIONS       HR    REGIONS         NO  YES     NO     NO
```

39.6.2 Listing the Materialized Views that Use a Materialized View Log

More than one materialized view can use a materialized view log.

If you have materialized view logs based at a master, then you can use the query in this section to list the following the materialized views that use each log:

- The name of each log table that stores the materialized view log data
- The owner of each materialized view log
- The master on which each materialized view log is based
- The materialized view identification number of each materialized view that uses the materialized view log
- The name of each materialized view that uses the materialized view log

To view this information, complete the following steps:

1. Connect to the master database as an administrative user.
2. Run the following query:

```
COLUMN LOG_TABLE HEADING 'Mview|Log Table' FORMAT A20
COLUMN LOG_OWNER HEADING 'Mview|Log Owner' FORMAT A10
COLUMN MASTER HEADING 'Master' FORMAT A20
COLUMN MVIEW_ID HEADING 'Mview|ID' FORMAT 9999
COLUMN NAME HEADING 'Mview Name' FORMAT A20
```

```
SELECT L.LOG_TABLE, L.LOG_OWNER, B.MASTER, B.MVIEW_ID, R.NAME
FROM ALL_MVIEW_LOGS L, ALL_BASE_TABLE_MVIEWS B, ALL_REGISTERED_MVIEWS R
WHERE B.MVIEW_ID = R.MVIEW_ID
AND B.OWNER = L.LOG_OWNER
AND B.MASTER = L.MASTER;
```

Your output looks similar to the following:

```
Mview                Mview                          Mview
Log Table            Log Owner  Master                  ID Mview Name
-------------------- ---------- -------------------- ----- --------------------
MLOG$_COUNTRIES      HR         COUNTRIES               21 COUNTRIES_MV1
MLOG$_DEPARTMENTS    HR         DEPARTMENTS             22 DEPARTMENTS_MV1
MLOG$_EMPLOYEES      HR         EMPLOYEES               23 EMPLOYEES_MV1
MLOG$_JOBS           HR         JOBS                    24 JOBS_MV1
MLOG$_JOB_HISTORY    HR         JOB_HISTORY             25 JOB_HISTORY_MV1
MLOG$_LOCATIONS      HR         LOCATIONS               26 LOCATIONS_MV1
MLOG$_REGIONS        HR         REGIONS                 27 REGIONS_MV1
```

39.7 Monitoring Materialized Views

You can run queries to display information about the materialized views and refresh groups.

- Listing Information About Materialized Views
  You can run queries to display information about the materialized views.
- Listing Information About the Refresh Groups at a Materialized View Database
  Each refresh group at a materialized view database is associated with a refresh job that refreshes the materialized views in the refresh group at a set interval.
- Determining the Job ID for Each Refresh Job at a Materialized View Database
  Query the `DBA_REFRESH` and `DBA_JOBS` views to determine the job identification number for each refresh job at a materialized view database.
- Determining Which Materialized Views Are Currently Refreshing
  Query the `V$MVREFRESH` view to determine which materialized views are currently refreshing.

39.7.1 Listing Information About Materialized Views

You can run queries to display information about the materialized views.

- Listing Master Database Information For Materialized Views
  Query the `DBA_MVIEWS` view to list the master database information for materialized views.
- Listing the Properties of Materialized Views
  Query the `DBA_MVIEWS` view to list the properties of materialized views.

39.7.1.1 Listing Master Database Information For Materialized Views

Query the `DBA_MVIEWS` view to list the master database information for materialized views.

Complete the following steps to show the master database for each materialized view at a replication database and whether the materialized view can be fast refreshed:

1. Connect to the materialized view database as an administrative user.
2. Run the following query:

```
COLUMN MVIEW_NAME HEADING 'Materialized|View Name' FORMAT A15
COLUMN OWNER HEADING 'Owner' FORMAT A10
```

```
COLUMN MASTER_LINK HEADING 'Master Link' FORMAT A30
COLUMN Fast_Refresh HEADING 'Fast|Refreshable?' FORMAT A16

SELECT MVIEW_NAME,
       OWNER,
       MASTER_LINK,
       DECODE(FAST_REFRESHABLE,
              'NO', 'NO',
              'DML', 'YES',
              'DIRLOAD', 'DIRECT LOAD ONLY',
              'DIRLOAD_DML', 'YES',
              'DIRLOAD_LIMITEDDML', 'LIMITED') Fast_Refresh
    FROM DBA_MVIEWS;
```

Your output looks similar to the following:

```
Materialized                                              Fast
View Name       Owner      Master Link                    Refreshable?
--------------- ---------- ------------------------------ ---------------
COUNTRIES_MV1   HR         @ORC1.EXAMPLE.COM              YES
DEPARTMENTS_MV1 HR         @ORC1.EXAMPLE.COM              YES
EMPLOYEES_MV1   HR         @ORC1.EXAMPLE.COM              YES
JOBS_MV1        HR         @ORC1.EXAMPLE.COM              YES
JOB_HISTORY_MV1 HR         @ORC1.EXAMPLE.COM              YES
LOCATIONS_MV1   HR         @ORC1.EXAMPLE.COM              YES
REGIONS_MV1     HR         @ORC1.EXAMPLE.COM              YES
```

39.7.1.2 Listing the Properties of Materialized Views

Query the `DBA_MVIEWS` view to list the properties of materialized views.

You can use the query in this section to list the following information about the materialized views at the current replication database:

- The name of each materialized view
- The owner of each materialized view
- The refresh method used by each materialized view: `COMPLETE`, `FORCE`, `FAST`, or `NEVER`
- The last date on which each materialized view was refreshed

To view this information, complete the following steps:

1. Connect to the materialized view database as an administrative user.
2. Run the following query to list this information:

To view this information, complete the following steps:

1. Connect to the materialized view database as an administrative user.
2. Run the following query:

```
COLUMN MVIEW_NAME HEADING 'Materialized|View Name' FORMAT A15
COLUMN OWNER HEADING 'Owner' FORMAT A10
COLUMN REFRESH_METHOD HEADING 'Refresh|Method' FORMAT A10
COLUMN LAST_REFRESH_DATE HEADING 'Last|Refresh|Date'
COLUMN LAST_REFRESH_TYPE HEADING 'Last|Refresh|Type' FORMAT A15

SELECT MVIEW_NAME,
       OWNER,
       REFRESH_METHOD,
       LAST_REFRESH_DATE,
```

```
   LAST_REFRESH_TYPE
FROM DBA_MVIEWS;
```

Your output looks similar to the following:

```
                                      Last      Last
Materialized                Refresh   Refresh   Refresh
View Name       Owner       Method    Date      Type
--------------- ----------- --------- --------- ---------------
COUNTRIES_MV1   HR          FAST      21-OCT-03 FAST
DEPARTMENTS_MV1 HR          FAST      21-OCT-03 FAST
EMPLOYEES_MV1   HR          FAST      21-OCT-03 FAST
JOBS_MV1        HR          FAST      21-OCT-03 FAST
JOB_HISTORY_MV1 HR          FAST      21-OCT-03 FAST
LOCATIONS_MV1   HR          FAST      21-OCT-03 FAST
REGIONS_MV1     HR          FAST      21-OCT-03 FAST
```

39.7.2 Listing Information About the Refresh Groups at a Materialized View Database

Each refresh group at a materialized view database is associated with a refresh job that refreshes the materialized views in the refresh group at a set interval.

You can query the `DBA_REFRESH` data dictionary view to list the following information about the refresh jobs at a materialized view database:

- The name of the refresh group.
- The owner of the refresh group.
- Whether the refresh job is broken.
- The next date and time when the refresh job will run.
- The current interval setting for the refresh job. The interval setting specifies the amount of time between the start of a job and the next start of the same job.

To view this information, complete the following steps:

1. Connect to the materialized view database as an administrative user.
2. Run the following query:

```
COLUMN RNAME HEADING 'Refresh|Group|Name' FORMAT A10
COLUMN ROWNER HEADING 'Refresh|Group|Owner' FORMAT A10
COLUMN BROKEN HEADING 'Broken?' FORMAT A7
COLUMN next_refresh HEADING 'Next Refresh'
COLUMN INTERVAL HEADING 'Interval' FORMAT A20

SELECT RNAME,
          ROWNER,
       BROKEN,
       TO_CHAR(NEXT_DATE, 'DD-MON-YYYY HH:MI:SS AM') next_refresh,
       INTERVAL
    FROM DBA_REFRESH
    ORDER BY 1;
```

Your output looks similar to the following:

```
Refresh    Refresh
Group      Group
Name       Owner      Broken? Next Refresh            Interval
---------- ---------- ------- ---------------------- --------------------
HR_REFG    MVIEWADMIN N       24-OCT-2003 07:18:44 AM SYSDATE + 1/24
```

The N in the Broken? column means that the job is not broken. Therefore, the refresh job will run at the next start time. A Y in this column means that the job is broken.

39.7.3 Determining the Job ID for Each Refresh Job at a Materialized View Database

Query the DBA_REFRESH and DBA_JOBS views to determine the job identification number for each refresh job at a materialized view database.

You can run a query to list the following information about the refresh jobs at a materialized view database:

- The job identification number of each refresh job. Each job created by Oracle Scheduler is assigned a unique identification number.
- The privilege schema, which is the schema whose default privileges apply to the job.
- The schema that owns each refresh job.
- The name of the refresh group that the job refreshes.
- The status of the refresh job, either normal or broken.

To view this information, complete the following steps:

1. Connect to the materialized view database as an administrative user.
2. Run the following query:

```
COLUMN JOB HEADING 'Job ID' FORMAT 999999
COLUMN PRIV_USER HEADING 'Privilege|Schema' FORMAT A10
COLUMN RNAME HEADING 'Refresh|Group|Name' FORMAT A10
COLUMN ROWNER HEADING 'Refresh|Group|Owner' FORMAT A10
COLUMN BROKEN HEADING 'Broken?' FORMAT A7

SELECT J.JOB,
       J.PRIV_USER,
       R.ROWNER,
       R.RNAME,
       J.BROKEN
    FROM DBA_REFRESH R, DBA_JOBS J
    WHERE R.JOB = J.JOB
    ORDER BY 1;
```

Your output looks similar to the following:

```
                  Refresh    Refresh
        Privilege Group      Group
 Job ID Schema    Owner      Name       Broken?
------- --------- ---------- ---------- -------
     21 MVIEWADMIN MVIEWADMIN HR_REFG    N
```

The N in the Broken? column means that the job is not broken. Therefore, the job will run at the next start time. A Y in this column means that the job is broken.

39.7.4 Determining Which Materialized Views Are Currently Refreshing

Query the V$MVREFRESH view to determine which materialized views are currently refreshing.

Complete the following steps to show the materialized views that are currently refreshing:

1. Connect to the materialized view database as an administrative user.

2. Run the following query:

```
COLUMN SID HEADING 'Session|Identifier' FORMAT 9999
COLUMN SERIAL# HEADING 'Serial|Number' FORMAT 999999
COLUMN CURRMVOWNER HEADING 'Owner' FORMAT A15
COLUMN CURRMVNAME HEADING 'Materialized|View' FORMAT A25

SELECT * FROM V$MVREFRESH;
```

Your output looks similar to the following:

```
   Session  Serial                 Materialized
Identifier  Number Owner           View
---------- ------- --------------- -------------------------
        19     233 HR              COUNTRIES_MV
         5     647 HR              EMPLOYEES_MV
```