MATERIALITZED VIEWS IN ORACLE 11G

This document summarizes two Oracle manuals, namely:

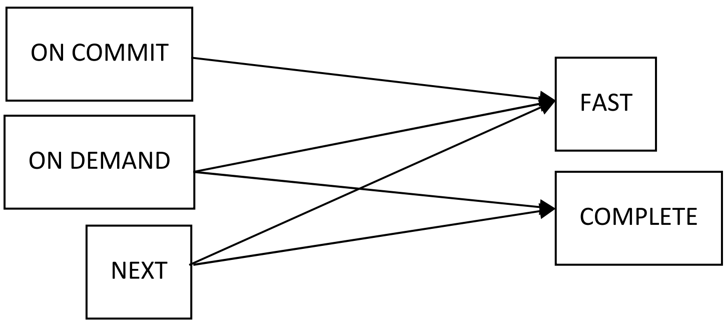
- The SQL reference manual (sections from 15-4 to 15-34, which describe how to create materialized views and logs to support incremental materialized views and
- The Data Warehousing Guide (mainly, sections 9 and 10, how to use materialized views to improve performance).

In this document you will find references (in brackets) to these two manuals. Follow them to gain insights or find examples about what is discussed here.

Before starting to deal with materialized views, remember a couple of restrictions any view (materialized or not) must fulfill:

- Views in general and materialized views in particular do not accept ORDER BY clause.
- If more than one attribute in the SELECT clause has the same name, the view definition requires to provide the attribute names (it is not enough to define alias for them in the SELECT clause) to allow query rewriting (see SQL Reference 15-21).
- When the list of columns in the SELECT has some kind of ambiguity, ALIAS are needed (see Data Warehousing Guide 9-15).

Now, let us focus on materialized views. When a materialized view is refreshed on commit, the time required to complete the commit may be slightly longer than usual. This is because the refresh operation is performed as part of the commit process. Therefore this method may not be suitable if many users are concurrently changing the tables upon which the materialized view is based. Periodical or on demand refresh can be either complete or incremental. Nevertheless, refresh on commit is only possible incrementally.



Any materialized view can always be completely refreshed. However, for a materialized view to be eligible for incremental update (what Oracle calls FAST refresh), you must define a log for each and every source table[1]. Moreover, both the query defining the view (from now on Q') and the log(s) created must satisfy some constraints depending on how Q' looks like. Below you can find a summary of the constraints to be fulfilled by the log(s) and Q' according to whether Q' does not contain neither joins nor grouping, if it contains joins **or**[2] if it contains grouping:

[1] Note that logs are not needed if the view is completely rebuilt (i.e., specifying complete as refresh parameter).

[2] You can have both joins and grouping at the same time but then, you must fulfill the constraints specified at both subsections.

<u>Constraints to fulfill when Q' is a basic materialized view (i.e., without joins nor grouping)</u>

- Characteristics to be fulfilled by Q'[3] (see Data Warehousing Guide 9-20):
  - You can use subqueries with EXISTS in the WHERE clause of Q' if the primary key of the subquery is joined to some attribute of the main query and the subquery table is not present in the main query.
  - It cannot contain nested queries that have IN, ANY, ALL, NOT IN or NOT EXISTS.
  - It cannot contain analytical functions (for example, RANK) in the SELECT clause.
  - It cannot have a HAVING clause with a subquery.
  - UNION ALL can only be in the main query, and not in the subqueries. To have it, ROWID must be projected together with a different constant for each query block of the UNION ALL (see Data Warehousing Guide 9-23).
  - If the FROM clause of the defining query references another materialized view, then you must always refresh the materialized view referenced in the defining query before refreshing the materialized view you are creating in this statement.

- Characteristics of the log (SQL Reference 15-31 and 15-32)
  The log definition tells you what will be recorded for each change in the source (or master) table. A log can record the following:
  - WITH PRIMARYKEY, ROWID, SEQUENCE (list_of_attrs)
    - SEQUENCE: Specify the SEQUENCE clause if the table is expected to have a mix of inserts, deletes, and updates (see Data Warehousing Guide 9-20).
    - WITH: The WITH clause can select PRIMARY KEY, ROWID, SEQUENCE and (list_of_attrs) at the same time but none of them can be used twice for the same materialized view (see SQL Reference 15-31). Bear in mind that if a list of attributes is provided, it does not have to be separated by ',' from any of the other three.
  - INCLUDING / EXCLUDING NEW VALUES
    - It tells Oracle to store (or do not store) old values after DML operations have been performed.

  As general rule, what to choose for each log depends on how Q' looks like, but keep in mind the following restrictions that must always hold in any log:
  - Any log must contain either PRIMARY KEY or ROWID (by default the former).
    - If the WITH PRIMARY KEY option is chosen, the primary key columns cannot appear in the list of attributes, if also selected (see SQL Reference 15-32).

  - Use "list_of_attrs" for attributes in the WHERE clause of the subquery, if any.

[3] Check the SQL-Reference manual (section 15-19) to know how to create a primary key or a rowid for the materialized view tuples.

Additional constraints to be fulfilled when Q’ contains joins

- Characteristics of Q’ (Data Warehousing Guide 9-21):
  - Query
    - The SELECT clause must contain the RowIDs of all tables in the FROM clause (see Data Warehousing Guide 8-11)[4].
    - Subqueries are not allowed.
    - Joins are not accepted in the FROM clause. Links between tables must be stated in the WHERE clause (use Oracle proprietary syntax for outer joins).
- Characteristics of the log (Data Warehousing Guide 9-21)
  - WITH ROWID
    - It must always appear.

Additional constraints to be fulfilled when Q’ contains grouping

- Characteristics of Q’ (Data Warehousing Guide 9-22)
  - Query
    - The SELECT clause must contain all GROUP BY columns.
    - Only SUM, COUNT, AVG, STDDEV, VARIANCE, MAX and MIN are supported for fast refreshed.
    - Nested aggregation functions, such as AVG(AVG(X)) are not allowed.
    - It must contain COUNT(*). Otherwise, fast refresh is supported only on inserts (such a materialized view is called an **insert-only materialized view**).
    - For any aggregate AGG(expr) (except MIN and MAX), it must contain COUNT(expr). Otherwise, it is an insert-only materialized view.
    - A materialized view with MAX or MIN is fast refreshable after delete or mixed DML statements if it does not have a WHERE.
    - The query cannot have a HAVING clause.
    - Check the Data Warehousing Guide 9-22 for using CUBE, ROLLUP and GROUPING SET in a materialized view.
- Characteristics of the log (Data Warehousing Guide 9-22)
  - INCLUDING NEW VALUES
    - It must always appear.
  - WITH ROWID
    - It must always appear.
  - SEQUENCE
    - It must always appear.
  - List of attributes
    - All attributes used in the materialized view must be recorded.

[4] Not necessary if the query also contains GROUP BY clause.

The FORCE refresh keyword

Specify FORCE to indicate that the materialized view can only be FAST refreshed partially. Oracle will take advantage of that part being FAST refreshable and will perform a COMPLETE refresh for the remaining. If you do not specify a refresh method (FAST, COMPLETE, or FORCE), then FORCE is the default (see Data Warehousing Guide 9-18 and 9-25).

Additional features provided by Oracle

Bear in mind that there is a specific Oracle package to work with materialized views. Two interesting methods are:

- To refresh the view on demand
  ```
  BEGIN DBMS_MVIEW.REFRESH('materialized_view_name'); END;
  /
  ```
- To check whether the view can be incrementally updated, proceed as follows:
  - Create the MV_CAPABILITIES_TABLE (see script bellow),
  - Create your log(s) and materialized view (but as REFRESH COMPLETE ON DEMAND),
  - Call the following procedure:
    ```
    BEGIN DBMS_MVIEW.EXPLAIN_MVIEW('materialized_view_name'); END;
    /
    ```
  - The previous call stores the explanations produced at the MV_CAPABILITIES_TABLE. To visualize them:
    ```
    SELECT * FROM MV_CAPABILITIES_TABLE
    ```
  - Among the explanations provided, look for those stating if FAST REFRESH is available or not. If it is not, Oracle suggests how to turn it into a FAST view in the comments field.

```sql
CREATE TABLE MV_CAPABILITIES_TABLE
(STATEMENT_ID      VARCHAR2(30),   -- Client-supplied unique statement identifier
 MVOWNER           VARCHAR2(30),   -- NULL for SELECT based EXPLAIN_MVIEW
 MVNAME            VARCHAR2(30),   -- NULL for SELECT based EXPLAIN_MVIEW
 CAPABILITY_NAME   VARCHAR2(30),   -- A descriptive name of the particular
                                  -- capability:
                                  -- REWRITE
                                  --   Can do at least full text match
                                  --   rewrite
                                  -- REWRITE_PARTIAL_TEXT_MATCH
                                  --   Can do at leat full and partial
                                  --   text match rewrite
                                  -- REWRITE_GENERAL
                                  --   Can do all forms of rewrite
                                  -- REFRESH
                                  --   Can do at least complete refresh
                                  -- REFRESH_FROM_LOG_AFTER_INSERT
                                  --   Can do fast refresh from an mv log
                                  --   or change capture table at least
                                  --   when update operations are
                                  --   restricted to INSERT
                                  -- REFRESH_FROM_LOG_AFTER_ANY
                                  --   can do fast refresh from an mv log
                                  --   or change capture table after any
                                  --   combination of updates
                                  -- PCT
                                  --   Can do Enhanced Update Tracking on
                                  --   the table named in the RELATED_NAME
                                  --   column.  EUT is needed for fast
                                  --   refresh after partitioned
                                  --   maintenance operations on the table
                                  --   named in the RELATED_NAME column
                                  --   and to do non-stale tolerated
                                  --   rewrite when the mv is partially
                                  --   stale with respect to the table
                                  --   named in the RELATED_NAME column.
                                  --   EUT can also sometimes enable fast
                                  --   refresh of updates to the table
                                  --   named in the RELATED_NAME column
                                  --   when fast refresh from an mv log
                                  --   or change capture table is not
                                  --   possible.
                                  -- See Table 8-7
 POSSIBLE          CHARACTER(1),  -- T = capability is possible
                                  -- F = capability is not possible
 RELATED_TEXT      VARCHAR2(2000), -- Owner.table.column, alias name, and so on
                                  -- related to this message. The specific
                                  -- meaning of this column depends on the
                                  -- NSGNO column. See the documentation for
                                  -- DBMS_MVIEW.EXPLAIN_MVIEW() for details.
 RELATED_NUM       NUMBER,        -- When there is a numeric value
                                  -- associated with a row, it goes here.
 MSGNO             INTEGER,       -- When available, QSM message # explaining
                                  -- why disabled or more details when
                                  -- enabled.
 MSGTXT            VARCHAR2(2000), -- Text associated with MSGNO.
 SEQ               NUMBER);       -- Useful in ORDER BY clause when
                                  -- selecting from this table.
```