8

Optimizing Joins with Join Groups

71

A **join group** is a user-created dictionary object that lists one or more columns that can be meaningfully joined.

This chapter contains the following topics:

8.1 About In-Memory Joins

Joins are an integral part of data warehousing workloads. The IM column store enhances the performance of joins when the tables being joined are stored in memory.

Because of faster scan and join processing, complex multitable joins and simple joins that use Bloom filters benefit from the IM column store. In a data warehousing environment, the most frequently-used joins involved a fact table and one or more dimension tables.

The following joins run faster when the tables are populated in the IM column store:

- Joins that are amenable to using Bloom filters
- Joins of multiple small dimension tables with one fact table
- Joins between two tables that have a primary key-foreign key relationship

8.2 About Join Groups

When the IM column store is enabled, the database can use join groups to optimize joins of tables populated in the IM column store.

A join group is a set of columns on which a set of tables is frequently joined. The column set contains one or more columns, with a maximum of 255 columns. The table set includes one or more internal tables. External tables are not supported.

The columns in the join group can be in the same or different tables. For example, if the `sales` and `times` tables frequently join on the `time_id` column, then you might create a join group for `(times(time_id), sales(time_id))`. If the `employees` table often joins to itself on the `employee_id` column, then a join group could be `(employees(employee_id))`.

> **Note:**
>
> The same column cannot be a member of multiple join groups.

When you create a join group, the database invalidates the current In-Memory contents of the tables referenced in the join group. Subsequent repopulation causes the database to re-encode the IMCUs of the tables with the common dictionary. For this reason, Oracle recommends that you first create the join group, and then populate the tables.

9 Optimizing Aggregation

71

Database In-Memory provides optimizations to speed up aggregation and arithmetic.

9.1 Optimizing In-Memory Aggregation with VECTOR GROUP BY

Starting with Oracle Database 12c Release 1 (12.1.0.2), **In-Memory Aggregation (IM aggregation)** enables queries to aggregate while scanning.

9.1.1 About IM Aggregation

IM aggregation optimizes query blocks involving aggregation and joins from a large table to multiple small tables.

The `KEY VECTOR` and `VECTOR GROUP BY` operations use efficient arrays for joins and aggregation. The optimizer chooses `VECTOR GROUP BY` for `GROUP BY` operations based on cost. The optimizer does not choose `VECTOR GROUP BY` aggregations for `GROUP BY ROLLUP`, `GROUPING SETS`, or `CUBE` operations.

> **Note:**
>
> IM aggregation is also called *vector aggregation* and *`VECTOR GROUP BY` aggregation*.

IM aggregation requires `INMEMORY_SIZE` to be set to a nonzero value. However, IM aggregation does not require that the referenced tables be *populated* in the IM column store.

> **See Also:**
>
> - "Enabling the IM Column Store for a Database"
> - *Oracle Database Data Warehousing Guide* to learn more about SQL aggregation

9.1.2 Purpose of IM Aggregation

IM aggregation preprocesses the small tables to accelerate the per-row work performed on the large table.