

ence Guide

Copyright Information

Introduction to SQL for Oracle NoSQL Database

Oracle NoSQL Database Data Model

Database Management

Index Management

Query Management

Row Management

Tables

Query Optimization

Using Indexes for Query Optimization

Finding Applicable Indexes

Examples: Using Indexes for Query Optimization

Optimizing unnesting queries with the UNNEST clause

Choosing the Best Applicable Index

Indexing

Query Plan

JSON Data Management

Inline Functions

Using Indexes for Query Optimization

46

In Oracle NoSQL Database, the query processor can identify which of the available indexes are beneficial for a query to make use of such an index. "Using" an index means scanning a contiguous subrange of its entries, potentially further filtering conditions on the entries within this subrange, and using the primary keys stored in the surviving entries to extract and return the associated table rows. The subrange of the index entries to scan is determined by the conditions in the WHERE clause, some of which may be converted to search conditions for the index. Given that only a (hopefully small) number of the index entries will satisfy the search conditions, the query can be evaluated without accessing each individual entry, saving a potentially large number of disk accesses.

Notice that in Oracle NoSQL Database, a primary-key index is always created by default. This index maps the primary key of a table to the physical location of the table rows. Furthermore, if no other index is available, the primary index is used. In other words, there is no pure "table scan" mechanism; a table scan is equivalent to a scan via the primary-key index.

When it comes to indexes and queries, the query processor must answer two questions:

1. Is an index applicable to a query? That is, will accessing the table via this index be more efficient than doing a full table scan (via the primary index).
2. Among the applicable indexes, which index or combination of indexes is the best to use?

Regarding question (1), for queries with NESTED TABLES, secondary indexes will be considered for the target table. In the current implementation, ancestor and/or descendant tables will always be accessed via their primary index.

Regarding question (2), the current implementation does not support index merging or index ordering. As a result, the query processor will always use exactly one index (which may be the primary-key index). Furthermore, there are no statistics on the distribution of values in a table column or nested fields. As a result, the query processor has to rely on some simple heuristics for choosing among the applicable indexes. In addition, SQL for Oracle NoSQL Database allows for the inclusion of HINTS in queries, which are used as user instructions to the query processor about which index to use.