Indexes Optimization in Oracle: (PERFORAMNCE TUNING)

Index optimization is one of the key performance tuning tasks for an Oracle DBA. It helps improve query performance by reducing the amount of data scanned during query execution. Let's break it down step by step.

1. **Why Optimize Indexes?**

- Optimizing indexes is crucial because:
- Speeds up queries by minimizing full table scans.
- Reduces I/O operations and CPU usage.
- Helps the optimizer choose the best execution plan.
- Avoids unnecessary indexes that waste space and degrade DML performance.

2. Types of Indexes in Oracle

Oracle provides different types of indexes for different use cases.

a) B-Tree Index (Balanced Tree Index)

- **Used for:** High-cardinality (unique or mostly unique values).
- **Structure:** A balanced tree structure that allows quick lookups.
- **Best for:** Primary keys, foreign keys, columns with many distinct values.
- **Optimization Steps:**

Use ANALYZE or DBMS_STATS to monitor index usage.

Rebuild fragmented indexes:

ALTER INDEX index_name REBUILD;

Avoid unnecessary indexes on frequently updated columns.

Use INDEX MONITORING to check if an index is used:

ALTER INDEX index_name MONITORING USAGE;

SELECT * FROM V$OBJECT_USAGE WHERE INDEX_NAME = 'index_name';

b) Bitmap Index

- **Used for:** Low-cardinality columns (few distinct values, e.g., Gender, Status).
- **Structure:** Stores bitmaps instead of row IDs, making queries faster.
- **Best for:** Read-heavy workloads, data warehouses, and columns with repetitive values.
- **Optimization Steps:**

1. Avoid using on transactional tables (DML operations on bitmap indexes cause row-level locking issues).
2. Use for OLAP queries and data warehousing.
3. Rebuild when fragmentation increases:

```
ALTER INDEX index_name REBUILD;
```

c) Function-Based Index (FBI)

- **Used for:** Queries using functions in the WHERE clause.
- **Example Problem:**

```
SELECT * FROM employees WHERE UPPER(last_name) = 'ALI';
```

Solution: Create a function-based index to avoid full table scans.

```
CREATE INDEX emp_upper_idx ON employees (UPPER(last_name));
```

- **Optimization Steps:**

1. Identify queries using functions on indexed columns.
2. Create an FBI only when function-based filtering is frequent.
3. Ensure optimizer can use the index by gathering statistics:

```
EXEC DBMS_STATS.GATHER_TABLE_STATS('HR', 'EMPLOYEES');
```

3. Steps to Optimize Index Usage

Step 1: Identify Unused Indexes

Check if an index is being used:

```
SELECT * FROM V$OBJECT_USAGE WHERE INDEX_NAME =
'index_name';
```

If an index is unused for a long time, consider dropping it:

```
DROP INDEX index_name;
```

Step 2: Rebuild Fragmented Indexes

Indexes can get fragmented due to frequent DML operations. Rebuild to improve performance:

```
ALTER INDEX index_name REBUILD;
```

Step 3: Check Index Selectivity

Use the DBA_IND_COLUMNS view to check column selectivity.

```
SELECT INDEX_NAME, COLUMN_NAME, DISTINCT_KEYS,
CLUSTERING_FACTOR
FROM DBA_IND_COLUMNS
WHERE TABLE_NAME = 'EMPLOYEES';
```

- High selectivity (many unique values) → Use B-Tree index.
- Low selectivity (few unique values) → Consider Bitmap index.

Step 4: Avoid Index Bloat

- Don't create indexes on small tables (full scans may be faster).
- Don't index columns with very low selectivity (e.g., Yes/No fields).
- Remove duplicate or redundant indexes.

Step 5: Use SQL Tuning Tools

Explain Plan: Check if the optimizer is using the index.

```
EXPLAIN PLAN FOR SELECT * FROM employees WHERE last_name = 'Ali';

SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

SQL Tuning Advisor:

```
EXEC DBMS_SQLTUNE.EXECUTE_TUNING_TASK(task_name => 'my_tuning_task');
```

AWR Reports: Check high I/O queries and missing indexes.

4. When to Avoid Indexing?

- Small tables (full table scans might be faster).
- High DML tables (frequent inserts/updates make index maintenance costly).
- Columns with very few distinct values (consider Bitmap index instead).

Final Takeaways

- Use B-Tree indexes for high-cardinality columns.
- Use Bitmap indexes for low-cardinality, read-heavy workloads.
- Use Function-Based Indexes when queries involve functions.
- Regularly monitor index usage, rebuild fragmented indexes, and drop unused indexes.
- Always check the execution plan to ensure the optimizer is using the index.

==========GOOD LUCK==========