Database and Storage Optimization

Overview

Database optimization enhances query performance, resource utilization, and overall reliability. These techniques are applicable to AWS-managed databases like Amazon RDS or Amazon DocumentDB.

Optimization Techniques

1. Indexing

Purpose: Improve data retrieval times by reducing the amount of data scanned during query execution.

Implementation: Identify columns frequently used in query WHERE or ORDER BY clauses and create indexes on them.

Example (PostgreSQL): Before Indexing:

sql

```
SELECT * FROM orders WHERE customer_id = '12345' ORDER BY
order_date DESC;
```

After Indexing:

sql

```
CREATE INDEX idx_orders_customer_date ON orders (customer_id,
order_date DESC);
```

Benefit: This composite index allows direct access to relevant rows, improving query efficiency.

2. Query Optimization

Purpose: Rewrite queries to eliminate redundant computations and optimize join logic.

Implementation: Use tools like PostgreSQL's EXPLAIN ANALYZE to identify performance bottlenecks.

Example: Before (Using Subquery):

sql

```
SELECT * FROM orders
```

```
WHERE customer_id IN (SELECT id FROM customers WHERE region =
'West');
```

After (Using JOIN):

sql
```
SELECT o.*
FROM orders o
INNER JOIN customers c ON o.customer_id = c.id
WHERE c.region = 'West';
```

Benefit: Better performance due to more efficient join algorithm selection.

3. Data Partitioning

Purpose: Divide large tables into smaller, more manageable pieces to improve performance.

Implementation: Partition data based on a specific key, such as date ranges or categorical values.

Example (PostgreSQL - Range Partitioning):

sql
```
-- Define the parent partitioned table
CREATE TABLE orders (
  order_id serial NOT NULL,
  customer_id int NOT NULL,
  order_date date NOT NULL,
  -- other columns
  PRIMARY KEY (order_id, order_date)
) PARTITION BY RANGE (order_date);

-- Create partitions
CREATE TABLE orders_2024 PARTITION OF orders
FOR VALUES FROM ('2024-01-01') TO ('2025-01-01');

CREATE TABLE orders_2025 PARTITION OF orders
FOR VALUES FROM ('2025-01-01') TO ('2026-01-01');
```

Benefit: Queries can scan only relevant partitions, significantly improving performance.