SQL Performance Optimization Tips

20 Best Practices for Optimizing SQL DDL, DQL, and DML Operations

PART 3

Dan Vukota

linkedin.com/in/danvukota/

DQL (Data Query Language) Optimization

17. Use EXISTS instead of IN for Subqueries

When the subquery involves a large dataset or a correlated subquery, EXISTS can be faster. This is because EXISTS checks for the existence of a row and stops as soon as it finds one, while IN requires checking all returned rows.

Example:

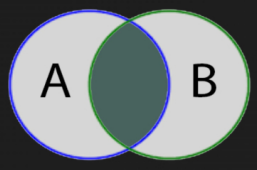
Use:

```
SELECT employee_id, name
FROM employees e
 WHERE EXISTS (
    SELECT 1
    FROM departments d
    WHERE d.department_id = e.department_id
    AND d.location = 'Toronto');
```

Instead of:

```
SELECT employee_id, name
FROM employees
WHERE department_id IN (
    SELECT department_id FROM departments
    WHERE location = 'Toronto');
```

18. Use INNER JOIN When Optimal



If you don’t need rows from the left table when there's no match in the right table, using an INNER JOIN can be more efficient than a LEFT JOIN. INNER JOIN only returns rows where a match exists, and databases can optimize it better.

Example:

Use:

```
SELECT e.employee_id, e.name
FROM employees e
INNER JOIN departments d ON
e.department_id = d.department_id
WHERE d.location = 'Toronto';
```

Instead of:

```
SELECT e.employee_id, e.name
FROM employees e
LEFT JOIN departments d ON
e.department_id = d.department_id
WHERE d.location = 'Toronto';
```

Avoid using DISTINCT in subqueries; instead, use an INNER JOIN when possible. DISTINCT can be costly, especially with large result sets, and an INNER JOIN can often eliminate duplicates before resorting to DISTINCT. Here is the same optimal query example using INNER JOIN compared to a variation using DISTINCT:

Use:

```
SELECT e.employee_id, e.name
FROM employees e
 INNER JOIN departments d ON
 e.department_id = d.department_id
 WHERE d.location = 'Toronto';
```

Instead of:

```
SELECT DISTINCT employee_id, name
FROM employees
WHERE department_id IN (
    SELECT department_id
    FROM departments WHERE
    location = 'Toronto');
```

DML (Data Manipulation Language) Optimization

19. Optimize UPDATE and DELETE

UPDATE and DELETE queries can be very expensive, especially on large tables. To optimize them, use the same best practices that apply to optimizing DQL operations.

Example #1:

```
UPDATE customers c
INNER JOIN orders o
ON c.customer_id = o.customer_id
SET c.status = 'Inactive'
WHERE COALESCE(c.status, 'None') = 'None'
  AND o.order_date >= '2023-01-01'
LIMIT 1000;
```

Example #2:

```
DELETE e
FROM employees e
INNER JOIN departments d
ON e.department_id = d.department_id
WHERE d.location = 'Lake Placid';
```

20. Avoid Triggers for Bulk Operations

While triggers can help enforce business rules, they can add significant overhead during UPDATE and DELETE operations, especially when multiple tables are involved. If you need to perform bulk updates or deletes, consider disabling triggers temporarily, if the integrity of your data is not at risk.

Thank You

Comment

Repost

Save

linkedin.com/in/danvukota/

Dan Vukota