4

Using Basic Techniques

Objectives

After completing this lesson, you should be able to:

- Describe how to develop efficient SQL statements
- Examine some common mistakes

Developing Efficient SQL Overview

There are several ways you can improve SQL statement efficiency:

- Verifying optimizing statistics
- Reviewing the execution plan
- Restructuring the inefficient SQL statements
- Restructuring the indexes
- Modifying or disabling triggers and constraints
- Restructuring the data
- Maintaining stable execution plans over time
- Visiting data as few times as possible

Scripts Used in This Lesson

The following scripts are used to show the basic SQL tips through examples of the inefficient SQLs:

- `create_sqlt.sh`: Configure the demo environment.
- Demo scripts are located in the `/home/oracle/demo` directory.
  - Execute demo scripts and format the traced information by using `tkprof: $ tkprof <trace file name> <output> sys=no`
  - Review the output of the demo script: `demo<nn>_<mm>_output.txt`
- Unless specified, execute all scripts as `sqlt` (password: `oracle_4U`).

Example 1: Table Design

Index Information

- customers : `cust_postal_code_ix: cust_postal_code`
- postal_codes : `postal_codes_pk: code1 + code2`

```
SELECT p.town_name, c.cust_last_name
FROM customers c, postal_codes p
WHERE p.code1 = substr(c.cust_postal_code,1,2)
AND p.code2 = substr(c.cust_postal_code,3,3)
AND p.code1 = '67'
AND c.country_id = 52790;

Rows     Row Source Operation
-------  ---------------------------------------------------
  911  NESTED LOOPS  (cr=3401 pr=1150 pw=0 time=330288 us)
  911   TABLE ACCESS FULL CUSTOMERS (cr=1517 pr=1150 pw=0 time=189016 us)
  911   TABLE ACCESS BY INDEX ROWID POSTAL_CODES (cr=1884 pr=0 pw=0 …)
  911    INDEX UNIQUE SCAN POSTAL_CODES_PK (cr=973 pr=0 pw=0 time=43418 us)
```

Example 2: Index Usage

Index Information

- customers : `customers_pk : cust_id`

```
[A] SELECT cust_first_name, cust_last_name FROM customers
    WHERE cust_id = 1030

[B] SELECT cust_first_name, cust_last_name FROM customers
    WHERE cust_id <> 1030

[C] SELECT cust_first_name, cust_Last_Name FROM customers
    WHERE cust_id < 10

[D] SELECT cust_first_name, cust_last_name FROM customers
    WHERE cust_id < 10000

[E] SELECT cust_first_name, cust_last_name FROM customers
    WHERE cust_id between 70 AND 80
```

Example 3: Transformed Index

Index Information

- customers : `cust_credit_limit : cust_cust_credit_limit_idx`

```
[A] SELECT cust_id
    FROM customers
    WHERE cust_credit_limit*1.10 = 11000

[B] SELECT cust_id
    FROM customers
    WHERE cust_credit_limit = 3000/2

Row Source Operation
---------------------------------------------------
TABLE ACCESS FULL CUSTOMERS (cr=1846 pr=1453 pw=0 time=42903 us cost=406 …)
```

Example 4: Data Type Mismatch

Index Information

- customers : `cust_postal_code_idx : cust_postal_code`

```
describe customers
 Name                          Null?    Type
 ----------------------------- -------- ----------------------
 CUST_POSTAL_CODE              NOT NULL VARCHAR2(10)
 …
SELECT cust_street_address
FROM customers
WHERE cust_postal_code =  68054;

Rows  Row Source Operation
----  -------------------------------------------------
 193  TABLE ACCESS FULL CUSTOMERS (cr=1471 pr=1448 pw=0
       time=147876 us)
```

Example 5: NULL usage

Index Information

- customers : `cust_marital_status_idx : cust_marital_status`

```
CREATE INDEX cust_maritial_status_idx
ON customers(cust_marital_status);

SELECT  count(*)
FROM customers;

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=1457 pr=1434 pw=0 time=106195 us)
  17428   TABLE ACCESS FULL CUSTOMERS (cr=1457 pr=1434 pw=0
           time=575669 us)
```

Example 6: Tune the ORDER BY Clause

```
[A] SELECT cust_first_name , cust_last_name, cust_credit_limit
    FROM customers
    ORDER BY cust_credit_limit;

[B] SELECT cust_first_name, cust_last_name, cust_credit_limit
    FROM customers
    ORDER BY cust_id;

[C] SELECT cust_first_name, cust_last_name, cust_city
    FROM customers
    WHERE cust_city = 'Paris'
    ORDER BY cust_id;

[d] SELECT cust_first_name, cust_last_name, cust_city
    FROM customers
    WHERE cust_id < 200
    ORDER BY cust_id;
```

Example 7: Retrieve a MAX value

Index Information

- customers : `cust_cust_credit_limit_ix : cust_credit_limit`

```
[A] SELECT max(cust_credit_limit)
    FROM customers;

Row Source Operation
---------------------------------------------
SORT AGGREGATE (cr=2 pr=1 pw=0 time=1177 us)
  INDEX FULL SCAN (MIN/MAX) CUST_CUST_CREDIT_LIMIT_IX ..)

[B] SELECT max(cust_credit_limit+1000)
    FROM customers;

Row Source Operation
---------------------------------------------
SORT AGGREGATE (cr=2 pr=1 pw=0 time=1177 us)
  INDEX FULL SCAN (MIN/MAX) CUST_CUST_CREDIT_LIMIT_IX ..)

[C] SELECT max(cust_credit_limit*2)
    FROM customers;
```

Example 8: Retrieve a MAX value

Index Information

- sales : `sales_pk : time_id + prod__id + cust_id + channel_id`

```
SELECT *
FROM sales
WHERE time_id = (SELECT max(time_id)
                 FROM sales
                 WHERE prod_id = :prod_id
                 AND cust_id = :cust_id);

Correct Result:

 PROD_ID    CUST_ID    TIME_ID   CHANNEL_ID   PROMO_ID QUANTITY_SOLD AMOUNT_SOLD
---------- ---------- ---------  ---------- ---------- ------------- -----------
     115      11457   29-DEC-98          3        999             1       10.61
```

Example 9: Correlated Subquery

```
SELECT department_id, last_name, salary
FROM employees e1
WHERE salary > (SELECT AVG(salary)
                FROM employees e2
                WHERE e1.department_id = e2.department_id
                GROUP BY e2.department_id)
ORDER BY department_id;

Rows     Row Source Operation
-------  ---------------------------------------------------
     38  SORT ORDER BY (cr=14 pr=7 pw=0 time=0 us cost=26..)
     38   FILTER  (cr=14 pr=7 pw=0 time=2960 us)
    107    TABLE ACCESS FULL EMPLOYEES (cr=7 pr=6 pw=0..)
     11    SORT GROUP BY NOSORT (cr=7 pr=1 pw=0 time=0..)
    106     TABLE ACCESS BY INDEX ROWID EMPLOYEES (cr=7..)
    106      INDEX RANGE SCAN EMP_DEPARTMENT_IX (cr=3 ..)
```

Example 10: `UNION` and `UNION ALL`

Index Information

- customers : `cust_first_name_idx : cust_first_name`
  `cust_last_name_idx : cust_last_name`

```
SELECT cust_last_name
FROM customers
WHERE cust_city = 'Paris'
UNION
SELECT cust_last_name FROM customers
WHERE cust_credit_limit < 10000

Rows     Row Source Operation
-------  ---------------------------------------------------
    883  SORT UNIQUE (cr=2915 pr=0..cost=1016 size=535572..)
  44837   UNION-ALL  (cr=2915 pr=0 pw=0 time=79452 us)
     77    TABLE ACCESS FULL CUSTOMERS (cr=1458 pr=0..)
  44760    TABLE ACCESS FULL CUSTOMERS (cr=1457 pr=0..)
```

Example 11: Avoid Using `HAVING`

Index Information

- customers: `cust_cust_city_idx: cust_city`

```
[A] SELECT cust_city, avg(cust_credit_limit)
    FROM customers
    GROUP BY cust_city
    HAVING cust_city = 'Paris';

[B] SELECT cust_city, avg(cust_credit_limit)
    FROM customers
    WHERE cust_city = 'Paris'
    GROUP BY cust_city
```

Example 12: Tune the BETWEEN Operator

Index Information

- customers : `cust_country_state_city_ix : country_id + cust_state_province + cust_city`

```
SELECT cust_id, cust_first_name, cust_last_name,
cust_state_province
FROM customers
WHERE country_id between 52788 and 52790
AND cust_state_province like 'W%';

Rows     Row Source Operation
-------  ---------------------------------------------------
   1125  TABLE ACCESS FULL CUSTOMERS (cr=1532 pr=1453 pw=0
         time=73972 us)
```

Example 13: Tune a Star Query by Using the Join Operation

Index Information

- sales : `sales_pk : prod_id + cust_id + time_id + channel_id`
- products : `products_pk : prod_id`

```
SELECT /*+ index(s sales_pk) */ sum(amount_sold)
FROM sales s
WHERE prod_id BETWEEN 130 AND 150
AND cust_id BETWEEN 10000 AND 10100;

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=1385 pr=0 pw=0 time=93104 us)
    637   TABLE ACCESS BY GLOBAL INDEX ROWID SALES PARTITION: …)
    637    INDEX RANGE SCAN SALES_PK (cr=929 pr=0 pw=0 time=6561 us)(…)
```

Example 14: Tune the Join Order

General Rules

- Avoid a full table scan if it is more efficient to get the required rows through an index.
- Avoid using an index that fetches 10,000 rows from the driving table if you could instead use another index that fetches 100 rows.
- Choose the join order so as to join fewer rows to tables later in the join order.

```
SELECT info
FROM taba a, tabb b, tabc c
WHERE a.acol BETWEEN 100 AND 200
AND b.bcol BETWEEN 10000 AND 20000
AND c.ccol BETWEEN 10000 AND 20000
AND a.key1 = b.key1
AND a.key2 = c.key2;
```

Example 15: Test for Existence of Rows

Query

- Check only if there are customers who purchased a specific product from those who have a credit limit that is greater than 10000.

```
......
SELECT count(*) into :v_count
FROM sales s, customers c
WHERE s.cust_id = c.cust_id
AND prod_id = :prod_id
AND c.cust_credit_limit > 10000;
IF v_count > 0 THEN
......
Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=1561 pr=1441 pw=0 time=155233 us)
   1422   HASH JOIN  (cr=1561 pr=1441 pw=0 time=146010 us)
   4805    TABLE ACCESS FULL CUSTOMERS (cr=1458 pr=1441 …)
  19403    INDEX RANGE SCAN SALES_PK (cr=103 pr=0 pw=0 …)
```

Example 16: `LIKE '%STRING'`

Index Information

- customers : `cust_last_name_ix : cust_last_name`

```
SELECT cust_first_name, cust_last_name
FROM customers
WHERE cust_last_name like '%ing';

Rows     Row Source Operation
-------  ---------------------------------------------------
    635  TABLE ACCESS FULL CUSTOMERS (cr=1501 pr=1426 pw=0
           time=19839 us)
```

Example 17: Use Caution When Managing Views

Query

- Find employees in a specified state.

```
CREATE OR REPLACE VIEW emp_dept
AS
SELECT d.department_id, d.department_name, d.location_id,
       e.employee_id, e.last_name, e.first_name, e.salary,
       e.job_id FROM  departments d ,employees e
WHERE e.department_id (+) = d.department_id;

SELECT v.last_name, v.first_name, l.state_province
FROM locations l, emp_dept v
WHERE l.state_province = 'California'
AND   v.location_id = l.location_id (+);
```

Example 18: Create a New Index

Index Information

- Costs : `costs_pk : prod_id + time_id + promo_id + channel_id`

```
SELECT prod_id, time_id, promo_id, channel_id, unit_cost
FROM costs
WHERE prod_id = 120;

Rows     Row Source Operation
-------  ---------------------------------------------------
   1974  PARTITION RANGE ALL PARTITION: 1 28 (cr=743 pr=0
         pw=0 time=91505 us)
   1974   TABLE ACCESS FULL COSTS PARTITION: 1 28 (cr=743
           pr=0 pw=0 time=47925 us)
```

Example 19: Join Column and Index

Index Information

- customers : `customers_pk : cust_id`

```
SELECT cust_state_province, sum(s.amount_sold)
FROM sales s, customers c
WHERE s.cust_id = c.cust_id
AND c.cust_year_of_birth= 1988
GROUP BY cust_state_province;
Rows     Row Source Operation
-------  ---------------------------------------------------
     14  HASH GROUP BY (cr=10558 pr=8104 pw=0 time=2462605 us)
   1903   HASH JOIN  (cr=10558 pr=8104 pw=0 time=2342695 us)
     82    TABLE ACCESS FULL CUSTOMERS (cr=1457 …)
 918843    PARTITION RANGE ALL PARTITION: 1 28 (cr=9101 …)
 918843     TABLE ACCESS FULL SALES PARTITION: 1 28 (cr=9101…)
```

Example 20: Ordering Keys for Composite Index

Index Information

- customers : `cust_country_state_city_ix : country_id + cust_state_province + cust_city`

```
SELECT count(*)
FROM customers
WHERE country_id > 52772
AND cust_state_province = 'CA'
AND cust_city = 'Belmont';

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=30 pr=0 pw=0 time=1634 us)
     30   INDEX SKIP SCAN CUST_COUNTRY_STATE_CITY_IX (cr=30
          pr=0 pw=0 time=1702 us)
```

Example 21: Bitmap Join Index

Index Information

- sales : `sales_pk : prod_id + cust_id + time_id + channel_id + promo_id`
- products : `products_pk : prod_id`

```
SELECT sum(s.quantity_sold)
FROM sales s, products p
WHERE s.prod_id = p.prod_id
AND p.prod_subcategory = 'CD-ROM';

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=1613 pr=2 pw=0 time=1450183 us)
  82817   HASH JOIN  (cr=1613 pr=2 pw=0 time=1840273 us)
      6    TABLE ACCESS BY INDEX ROWID PRODUCTS (cr=2 pr=0 pw=0 …)
      6     INDEX RANGE SCAN PRODUCTS_PROD_SUBCAT_IX (cr=1 …)
 918843    PARTITION RANGE ALL PARTITION: 1 28 (cr=1611 pr=2 pw=0…)
 918843     TABLE ACCESS FULL SALES PARTITION: 1 28 (cr=1611 …)
```

Example 22: Tune a Complex Logic

Index Information

- categories : `cat_ix : prod_category_id + prod_subcat_seq`

```
......
SELECT max(prod_subcat_seq) + 1 into v_next_seq
FROM categories
WHERE prod_category_id = v_prod_category_id;
  IF sqlcode = 100 THEN
    insert into categories
    values (v_prod_category_id, 1, v_prod_subcategory);
  ELSE
    insert into categories
    values (v_prod_category_id, v_next_seq,
            v_prod_subcategory);
  END IF;
......
```

Example 23: Writing Combined SQL Statement

```
SELECT count(*)
FROM customers
WHERE cust_gender='F'
AND country_id=52771;

SELECT count(*)
FROM customers
WHERE cust_gender='F'
AND country_id=52771
AND cust_marital_status is not null;

SELECT count(*)
FROM customers
WHERE cust_gender='F'
AND country_id=52771
AND (cust_marital_status is null OR
cust_marital_status='single');
```

Example 24: Write a Multitable INSERT Statement

```
INSERT INTO sales VALUES (product_id, customer_id, today, 3, promotion_id,
quantity_per_day, amount_per_day)
SELECT TRUNC(s.sales_date) AS today, s.product_id, s.customer_id,
s.promotion_id, SUM(s.amount) AS amount_per_day, SUM(s.quantity)
quantity_per_day, p.prod_min_price*0.8 AS product_cost, p.prod_list_price
AS product_price
FROM sales_activity_direct s, products p
WHERE s.product_id = p.prod_id AND TRUNC(sales_date) = TRUNC(SYSDATE)
GROUP BY TRUNC(sales_date), s.product_id, s.customer_id,
s.promotion_id, p.prod_min_price*0.8,
p.prod_list_price;

INSERT INTO costs VALUES (product_id, today, promotion_id, 3,product_cost,
product_price)
SELECT TRUNC(s.sales_date) AS today, s.product_id, s.customer_id,
s.promotion_id, SUM(s.amount) AS amount_per_day, SUM(s.quantity)
quantity_per_day, p.prod_min_price*0.8 AS product_cost, p.prod_list_price
AS product_price
FROM sales_activity_direct s, products p
WHERE s.product_id = p.prod_id AND TRUNC(sales_date) = TRUNC(SYSDATE)
GROUP BY TRUNC(sales_date), s.product_id, s.customer_id,
s.promotion_id, p.prod_min_price*0.8,
p.prod_list_price;
```

Example 25: Using Temporary Table

```
[A] SELECT sum(amount_sold)
    FROM sales s, times t, customers c
    WHERE s.time_id  = t.time_id
    AND s.cust_id  = c.cust_id
    AND t.day_name = 'Friday'
    AND country_id = 52772;
[B] SELECT sum(amount_sold)
    FROM sales s, times t, products p
    WHERE s.time_id  = t.time_id
    AND s.prod_id  = p.prod_id
    AND t.day_name = 'Friday'
    AND prod_category = 'Electronics';
[C] SELECT sum(amount_sold)
    FROM sales s, times t, promotions p
    WHERE s.time_id  = t.time_id
    AND s.promo_id = p.promo_id
    AND t.day_name = 'Friday'
    AND promo_category= 'TV';
```

Example 26: Using the `WITH` Clause

```
[A] SELECT s.prod_id, s.amount_sold, t.week_ending_day
    FROM sales s , times t , products p
    WHERE s.time_id = t.time_id AND s.prod_id = p.prod_id
    AND p.prod_category = 'Photo'
    AND p.prod_name LIKE '%Memory%'
    AND t.week_ending_day BETWEEN TO_DATE('01-JUL-2001','dd-
    MON-yyyy')
    AND TO_DATE('16-JUL-2001','dd-MON-yyyy');
[B] SELECT p.prod_name product, s.week_ending_day,
    SUM(s.amount_sold) revenue FROM products p LEFT OUTER
    JOIN (SELECT prod_id, amount_sold, week_ending_day
    FROM sales_numbers) s ON (s.prod_id = p.prod_id)
    WHERE p.prod_category = 'Photo' AND p.prod_name LIKE
   '%Memory%'
    GROUP BY p.prod_name, s.week_ending_day
[C] SELECT distinct week_ending_day week FROM times
    WHERE week_ending_day BETWEEN TO_DATE('01-JUL-2001','dd-
    MON-yyyy') AND TO_DATE('16-JUL-2001','dd-MON-yyyy')
[D] SELECT w.week, pr.product, nvl(pr.revenue,0) revenue
    FROM product_revenue pr PARTITION BY (product) RIGHT
    OUTER JOIN weeks w ON (w.week = pr.week_ending_day)
```

Example 27: Using the Materialized View

```
-- Business user query 1
SELECT cust_last_name, SUM(amount_sold)
FROM sales s, customers c
WHERE s.cust_id = c.cust_id
GROUP BY cust_last_name;

-- Business user query 2
SELECT channel_id,
SUM(amount_sold)
FROM sales
GROUP BY channel_id;
```

Example 28: Star Transformation

Index Information

- sales : `sales_pk : prod_id + cust_id + time_id + channel_id + promo_id`
- products : `products_pk : prod_id`
- channels : `channels_pk : channel_id`
- customers : `customers_pk : cust_id`

```
SELECT s.amount_sold,p.prod_name,ch.channel_desc
FROM sales s, products p, channels ch, customers c
WHERE s.prod_id=p.prod_id
AND s.channel_id=ch.channel_id
AND s.cust_id=c.cust_id
AND ch.channel_id in (3, 4)
AND c.cust_city='Asten'
AND p.prod_id>100;
```

Example 29: Partition Pruning

Index Information

- sales_np : `sales_pk : prod_id + cust_id + time_id + channel_id + promo_id`

```
SELECT sum(quantity_sold)
FROM sales_np
WHERE time_id between to_date('19980101', 'yyyymmdd')
AND to_date('19981231', 'yyyymmdd');

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=4441 pr=4182 pw=0 …)
 178834   TABLE ACCESS FULL SALES_NP (cr=4441…)
```

Example 30: Using a Bind Variable

Index Information

- tab1 : `tab1_b_ix : b`

```
SELECT count(*) FROM tab1 WHERE a=1;   -> 1 row
SELECT count(*) FROM tab1 WHERE a=5;   -> 1 row
SELECT count(*) FROM tab1 WHERE a=100; -> 1 row

SELECT sql_text,executions
FROM v$sql
WHERE sql_text like '%tab1%';

SQL_TEXT                                        EXECUTIONS
----------------------------------------  ----------------
select count(*) from tab1 where a=1                    1
select count(*) from tab1 where a=5                    1
select count(*) from tab1 where a=100                  1
```

Summary

In this lesson, you should have learned to:

- Describe how to develop efficient SQL statements
- Examine some common mistakes

Practice 4: Overview

This practice covers the following topics:

- Rewriting queries for better performance
- Rewriting applications for better performance