SQL Optimization

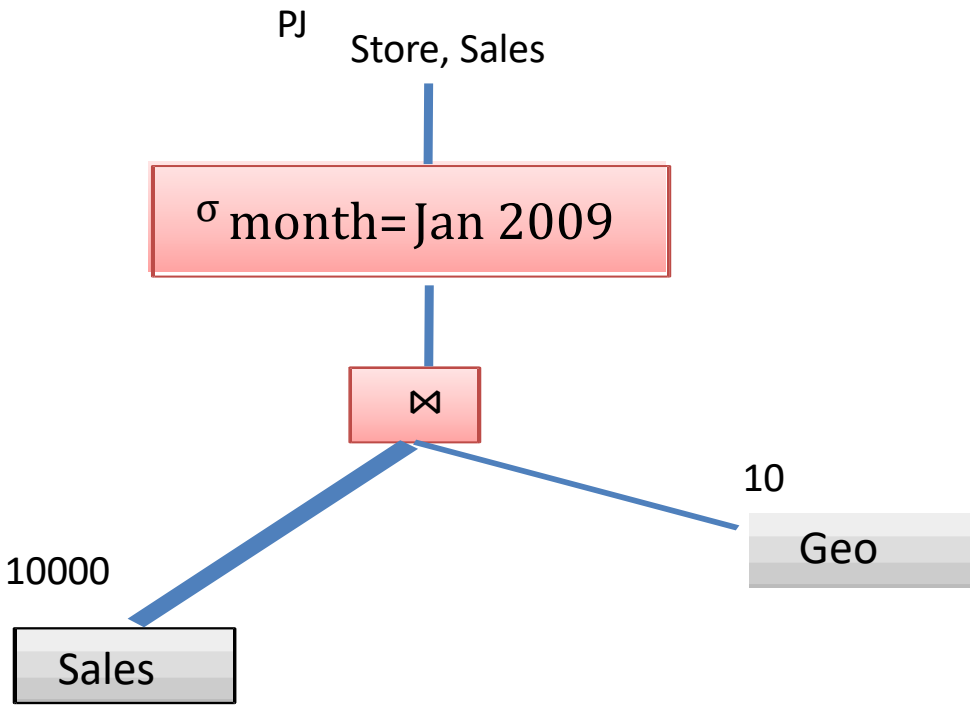
INFS602 Physical Database Design

Learning Outcomes

- Identify and re-write inefficient SQL queries

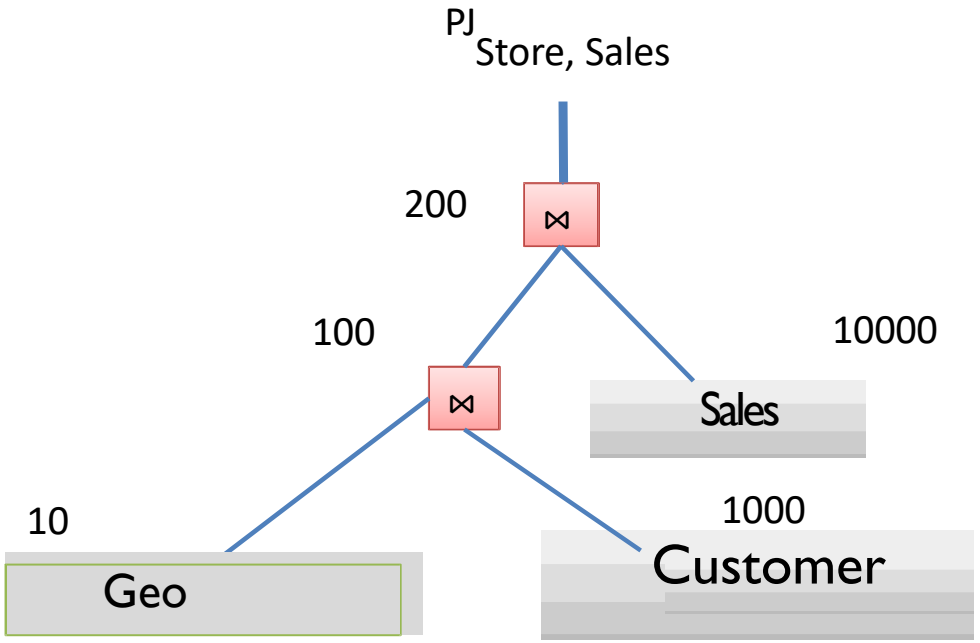
SQL Tuning Tips

- Select the most efficient driving table
- Two Table Joins
  - use the smaller table as the driver



SQL Tuning Tips

- Three Table Joins
  - use the pair that gives the smallest number of rows for the intermediate result



SQL Tuning

2. Use joins instead of *EXISTS*

```
Select  *
From Sales S
Where exists  (Select *
                  From Product P
                  Where S.prod_id=P.prod_id and P.qoh=4);
```

– 0.17 seconds

is less efficient than

```
Select *
From Sales S, Product P
Where S.prod_id=P.prod_id and P.qoh=4;
```

– 0.03 seconds

SQL Tuning

3. Use *EXISTS* in place of *DISTINCT*

- Avoid joins that require the DISTINCT qualifier on the SELECT list when you submit queries used to determine information at the owner end of a 1-many relationship

```
Select distinct P.prod_id, P.description from product P, sales S
where P.prod_id=S.prod_id
```

- 0.15 seconds

is less efficient than

```
Select P.prod_id, P.description from product P
where exists
        (select * from sales S
            where P.prod_id=S.prod_id)
```

- The optimizer realises that the subquery can be terminated when the query has been satisfied once

- 0.05 seconds

SQL Tuning

5. Use UNION in place of OR

```
SELECT sale_id, prod_id, cust_id from sales
WHERE prod_id=300140 OR prod_id=500390 OR
        prod_id=200340
```

-0.719 seconds

- Rewrite the above as:

```
SELECT sale_id, prod_id, cust_id from sales
    WHERE prod_id=300140
UNION
SELECT sale_id, prod_id, cust_id from sales
    WHERE prod_id=500390
UNION
SELECT sale_id, prod_id, cust_id from sales
    WHERE prod_id=200340
```

-0.328 seconds

SQL Tuning

6. Avoid calculations on indexes

```
SELECT *
FROM Emp
WHERE salary*12>25000;
```

- 0.15 seconds

- In this case Oracle will ignore any index that is defined on the salary column
- To take advantage of such an index re-write as:

```
SELECT *
FROM Emp
WHERE salary>25000/12;
```

- 0 seconds

SQL Tuning

7. Avoid WHERE constructs that suppress use of indexes

e.g. WHERE substr(origin,1,4) like 'AUCK';

```
Rows     Row Source Operation
-------  ---------------------------------------------------
9008  TABLE ACCESS FULL FLIGHTBKUP
```

- Replace with:

WHERE origin like ‘AUCK%'

```
Rows     Row Source Operation
-------  ---------------------------------------------------
9008  INDEX RANGE SCAN FLIGHT_ORIGIN_IDX
```

SQL Tuning

- If possible, rewrite subqueries that use an IN clause (replace with a straight join)

```
SELECT ename, sal
FROM emp
WHERE deptno IN
                    (SELECT deptno
                    FROM dept
                    WHERE loc like 'AUCK%');
```

Performance Comparison of Oracle's Execution Strategies for some selected Query Types

SQL QUERY	Query Version	Disk Reads	CPU Time	Elapsed Time
select * from sales s2, products p2 where s2.prod_id=p2.prod_id AND p2.qoh=4	Efficient	53	20000	1.83
select * from sales s3 where exists (select * from products p3 where s3.prod_id = p3.prod_id AND p3.qoh = 4)	Inefficient	53	40000	2.01

Performance Comparison of Oracle's Execution Strategies for some selected Query Types

SQL QUERY	Query Version	Disk Reads	CPU Time	Elapsed Time
select P.prod_id, P.description from products P where exists (select * from sales S where P.prod_id=S.prod_id)	Efficient	28	10000	2.04
select distinct P.prod_id, P.description from products P, sales S where P.prod_id=S.prod_id	Inefficient	28	30000	2.93

Performance Comparison of Oracle's Execution Strategies for some selected Query Types

SQL QUERY	Query Version	Disk Reads	CPU Time	Elapsed Time
select sale_id, prod_id, cust_id from sales where prod_id =300140 union select sale_id, prod_id, cust_id from sales where prod_id =500390 union select sale_id, prod_id, cust_id from sales where prod_id=200340	Efficient	7	10000	1.10
select sale_id, prod_id, cust_id from sales where prod_id=300140 OR prod_id=500390 OR prod_id=200340	Inefficient	54	20000	1.80