Exploring Best Practices and Guidelines for Tuning SQL Statements

Ami Aharonovich

Oracle ACE & OCP

Ami@DBAces.co.il

DBAces

Israel Oracle User Group

Who am I

- Oracle ACE 

- Oracle Certified Professional DBA (OCP) 

- Founder and CEO, **DBAces**
- Oracle DBA consultant and instructor, specializes in providing professional services and delivering Oracle trainings dealing with Oracle database core technologies and features
- Frequent speaker at the Oracle Open World annual event and various user group conferences around the globe
- President, Israel Oracle User Group 

DBAces

Israel Oracle User Group

Agenda

- Parsing, Bind Peeking and Adaptive Cursor Sharing
- Using EXISTS Operator and the WITH Clause
- Writing Scalar Subqueries
- Oracle Database 11g SQL Query Result Cache
- Indexes and Nulls and Invisible Index
- Oracle Database 12c New Features:
  - Adaptive Query Optimization
  - Enhanced Statistics

DBAces

Israel Oracle User Group

Parsing Time is Important

```
BEGIN
   FOR i IN 1..100000 LOOP
   EXECUTE IMMEDIATE 'INSERT INTO t (x,y)
                      VALUES ('||i||',''A'')';
   END LOOP;
END;
/
```

PL/SQL procedure successfully completed.

Elapsed: 00:00:42.91

Parsing_Time_is_Important.sql

DBAces

Israel Oracle User Group

Parsing Time is Important

```
BEGIN
   FOR i IN 1..100000 LOOP
   EXECUTE IMMEDIATE 'INSERT INTO t (x,y)
                      VALUES (:i,''A'')' USING i;
   END LOOP;
END;
/
```

PL/SQL procedure successfully completed.

Elapsed: 00:00:08.26

DBAces

il OUG
Israel Oracle User Group

Parsing Time is Important

```
SELECT SUBSTR(sql_text,11,8) "Bind", COUNT(*),
       ROUND(SUM(sharable_mem)/1024) "Memory KB"
FROM   v$sqlarea
WHERE  sql_text LIKE 'INSERT%INTO t (x,y)%'
GROUP BY SUBSTR(sql_text,11,8);
```

Bind	COUNT(*)	Memory KB
NO_BIND	9,349	131,580
USE_BIND	1	14

DBAces

Israel Oracle User Group

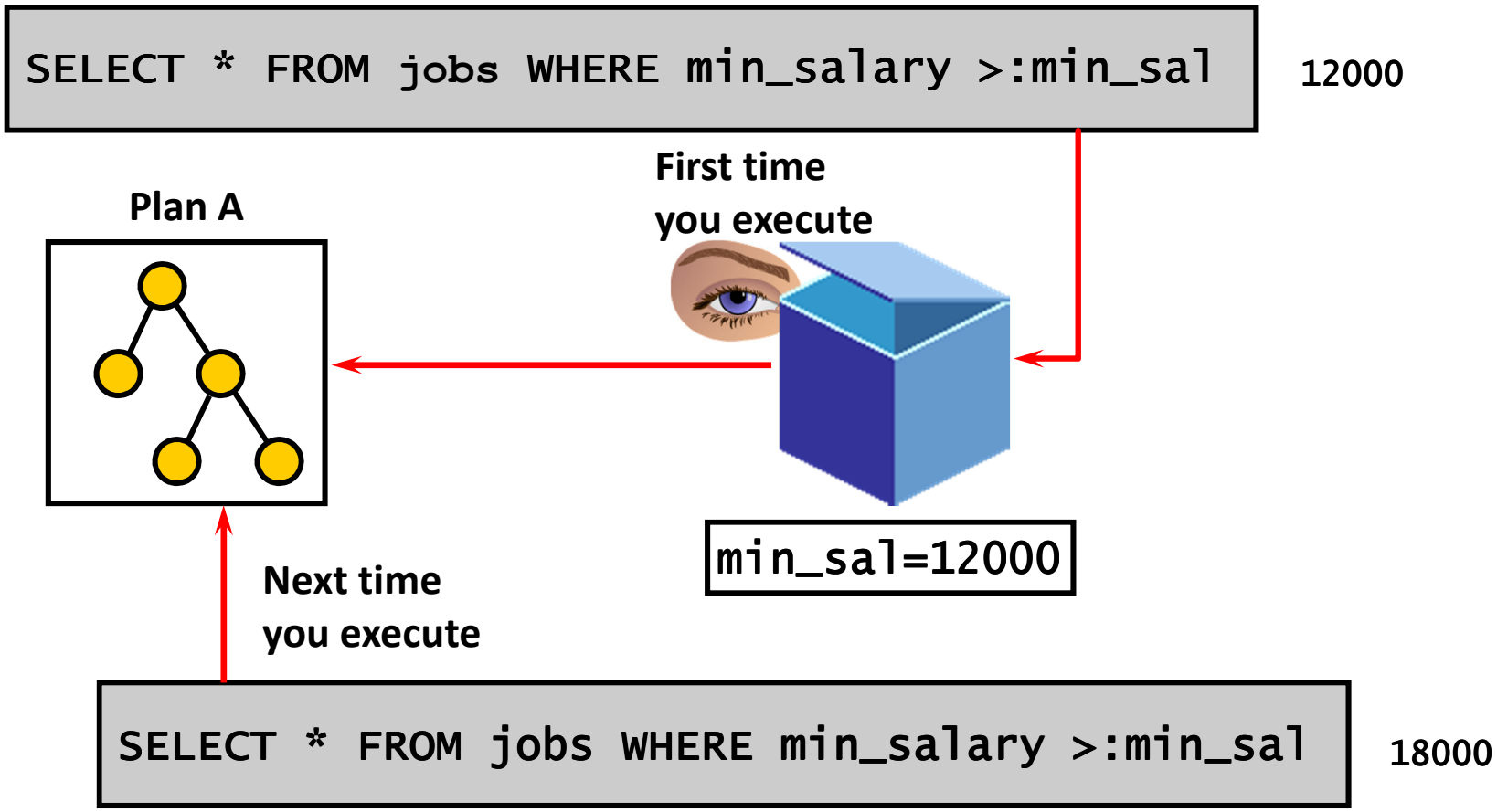
Bind Variable Peeking

- when the query is first ***hard parsed***, the optimizer will peek at the binds in order to determine how to optimize the query
- Lets the optimizer determines the selectivity
- Since Oracle9i (9.0.1)
- _optim_peek_user_binds=false

DBAces

Israel Oracle User Group

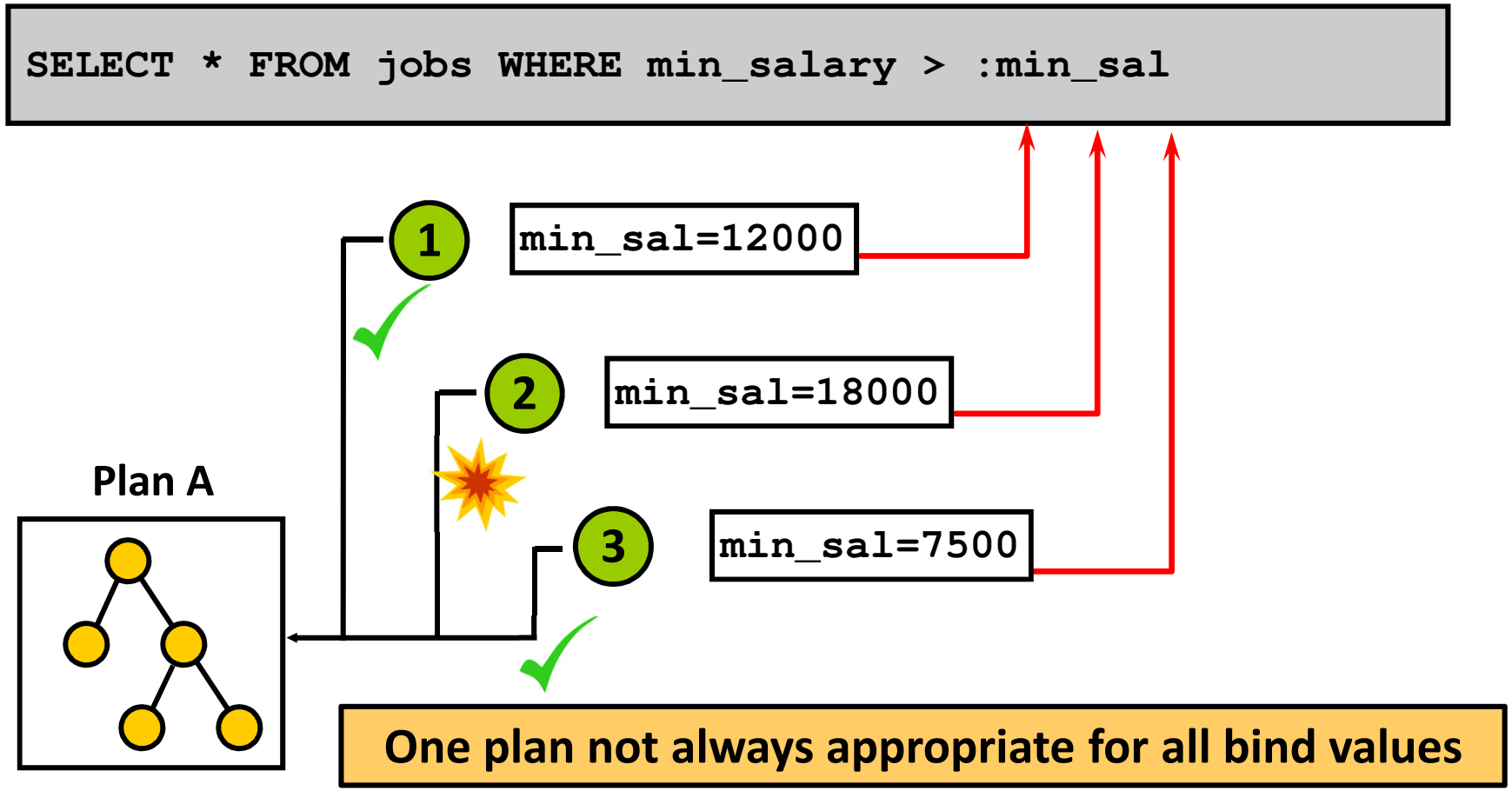
Bind Variable Peeking



DBAces

il OUG
Israel Oracle User Group

Bind Variable Peeking



DBAces

Israel Oracle User Group

11g Adaptive Cursor Sharing

- ***One plan not always appropriate for all bind values***
- Provides intelligent cursor sharing for statements that use bind variables and allows the optimizer to generate a set of plans that are optimal for different sets of bind values
- Adaptive Cursor Sharing benefits:
  - Automatically detects when different executions would benefit from different execution plans
  - Limits the number of generated child cursors to a minimum
  - Automated mechanism that cannot be turned off

Adaptive_Cursor_Sharing.sql

DBAces

il OUG
Israel Oracle User Group

Inefficient SQL: Examples

1
```
SELECT p.prod_id,p.prod_name,p.prod_list_price
FROM SH.products p
WHERE p.prod_list_price > 5 * (SELECT AVG(unit_cost)
FROM SH.costs c WHERE c.prod_id = p.prod_id);
```

2
```
SELECT * FROM job_history jh, employees e
WHERE substr(to_char(e.employee_id),2) =
      substr(to_char(jh.employee_id),2);
```

3
```
SELECT * FROM orders WHERE order_id_char = 1205;
```

4
```
SELECT * FROM employees WHERE to_char(salary) = :sal;
```

5
```
SELECT * FROM parts_old UNION SELECT * FROM parts_new;
```

DBAces

Israel Oracle User Group

Which One is Better ?

```
SELECT p.prod_id , p.prod_name , p.prod_list_price
FROM SH.products p
WHERE p.prod_list_price > 5 * (SELECT AVG(unit_cost)
FROM SH.costs c
WHERE c.prod_id = p.prod_id);
```

Cost = 222K / Elapsed = 00:00:01.38

```
SELECT p.prod_id , p.prod_name , p.prod_list_price
FROM SH.products p,
     (SELECT prod_id , AVG(unit_cost) avg_cost
      FROM SH.costs GROUP BY prod_id) c
WHERE p.prod_id = c.prod_id
AND   p.prod_list_price > 5 * c.avg_cost;
```

Cost = 991 / Elapsed = 00:00:00.26

Which_One_is_Better.sql

DBAces

Israel Oracle User Group

WITH Clause

- Define a query block before using it in a query
- Use the same query block in a select statement when it occurs more than once within a complex query
- Particularly useful when a query has many references to the same query block and there are joins and aggregations

DBAces

Israel Oracle User Group

WITH Clause Benefits

- Makes the query easier to read
- Evaluates a clause only once, even if it appears multiple times in the query
- In most cases, may improve performance for large queries

Which_One_is_Better_2.sql

DBAces

il OUG
Israel Oracle User Group

Use a Join – Not Always…

- In general, you use a join when you need data from more than one table in the ultimate SELECT list
- When you need data only from one table, it is unlikely that you would consider a join. Instead consider using either WHERE EXISTS or WHERE IN

```
SELECT DISTINCT d.dname,d.loc
FROM emp e,dept d WHERE e.deptno=d.deptno;
```

```
SELECT d.dname,d.loc FROM dept d
WHERE EXISTS ( SELECT 1 FROM emp e
               WHERE e.deptno = d.deptno);
```

DBAces

Israel Oracle User Group

Using the EXISTS Operator

- Used to test for existence of rows in the result set of the subquery
- Ensures that the search in the inner query **does not** continue when at least one match is found
- Should be used with correlated subqueries to test whether a value retrieved by the outer query exists in the results set of the values retrieved by the inner query

DBAces

Israel Oracle User Group

Scalar Subqueries

- Subquery that returns exactly one column value from one row
- Allows you to embed SQL statements which return a scalar value within SQL statements
- Can be used in:
  - The condition and expression part of `DECODE`/`CASE`
  - All clauses of `SELECT` except `GROUP BY`
  - The `SET` clause and `WHERE` clause of an `UPDATE` statement

DBAces

Israel Oracle User Group

Scalar Subqueries Caching

- Most important (and surprising) advantage of using scalar subqueries
- Oracle caches the results of a scalar subquery and reuses the value, even though the query should be called more than once

Scalar_Subquery.sql

DBAces

Israel Oracle User Group

SQL Query Result Cache

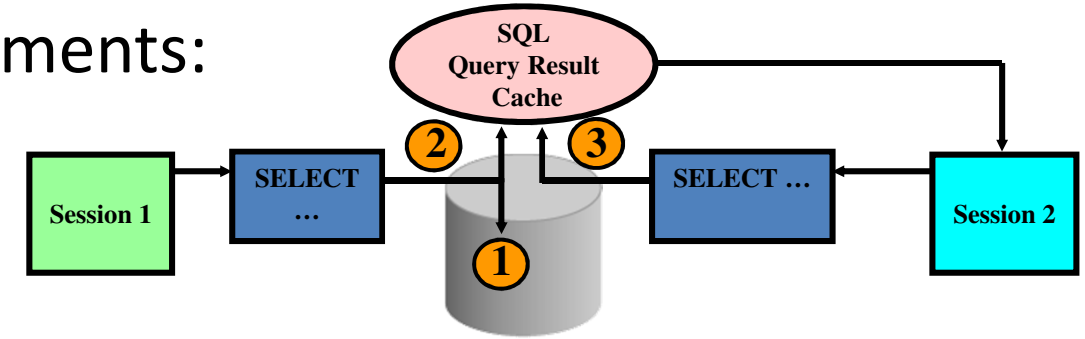
- In the past, the database always cached blocks of data, the building blocks used to build result sets
- Starting with Oracle Database 11*g*, the database can now also cache result sets!
- If you have a query that is executed over and over again against slowly or never-changing data, you will find the new server results cache to be of great interest
- This is a feature from which virtually every application can and will benefit

DBAces

il OUG
Israel Oracle User Group

SQL Query Result Cache: Overview

- A dedicated memory buffer stored in the shared pool is used for storing and retrieving the cached results
- The query results become invalid when data in the objects being accessed by the query is modified
- Multiple users can see these results without repeating the same query
- Good candidate statements:
  - Access many rows
  - Return few rows



SQL_Query_Result_Cache.sql

DBAces

il OUG
Israel Oracle User Group

Indexes and Nulls

```
SELECT * FROM employees WHERE department_id = 90;

-----------------------------------------------------------
| Id  | Operation                    | Name                 |
-----------------------------------------------------------
|   0 | SELECT STATEMENT             |                      |
|   1 |  TABLE ACCESS BY INDEX ROWID| EMPLOYEES            |
|*  2 |   INDEX RANGE SCAN           | EMP_DEPARTMENT_IX    |
-----------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("DEPARTMENT_ID"=90)
```

DBAces

Israel Oracle User Group

Indexes and Nulls

```
SELECT * FROM employees WHERE department_id IS NULL;

-------------------------------------------
| Id  | Operation           | Name        |
-------------------------------------------
|   0 | SELECT STATEMENT    |             |
|*  1 |  TABLE ACCESS FULL  | EMPLOYEES   |
-------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter("DEPARTMENT_ID" IS NULL)
```

DBAces

il OUG
Israel Oracle User Group

Indexes and Nulls

```
CREATE INDEX dept_ix_with_nulls ON employees(department_id,0);

SELECT * FROM employees WHERE department_id IS NULL;

-------------------------------------------------------------
| Id  | Operation                     | Name                  |
-------------------------------------------------------------
|   0 | SELECT STATEMENT              |                       |
|   1 |  TABLE ACCESS BY INDEX ROWID| EMPLOYEES             |
|*  2 |   INDEX RANGE SCAN            | DEPT_IX_WITH_NULLS    |
-------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("DEPARTMENT_ID" IS NULL)
```

DBAces

Israel Oracle User Group

Nulls and Cardinality and Indexes

- There is a pervasive myth that indexes and NULLs are like matter and anti-matter
- There is the though that “WHERE COLUMN IS NULL” cannot use an index
- There is a thought that NULLs cannot be indexed
- *“So lets use invalid values instead of the NULLs”*

Nulls_and_Cardinality.sql

DBAces

Israel Oracle User Group

Oracle11g Invisible Index: Overview

- Invisible indexes are ignored by the optimizer unless you explicitly set `OPTIMIZER_USE_INVISIBLE_INDEXES` initialization parameter to `TRUE` (default is `FALSE`)
- Alternative to making it unusable or dropping it
- Using invisible indexes you can:
  - Test the removal of an index before dropping it
  - Use temporary index structures for certain operations without affecting the overall application
- Invisible index are maintained during DML statements

DBAces

il OUG
Israel Oracle User Group

Invisible Indexes: Examples

- Index is altered as not visible to the optimizer:

```
ALTER INDEX ind1 INVISIBLE;
```

- Optimizer does not consider this index:

```
SELECT /*+ index(TAB1 IND1) */ COL1 FROM TAB1 WHERE …;
```

- Optimizer will always consider the index:

```
ALTER INDEX ind1 VISIBLE;
```

- Creating an index as invisible initially:

```
CREATE INDEX IND1 ON TAB1(COL1) INVISIBLE;
```

DBAces

il OUG
Israel Oracle User Group

You Are Being Watched

- We watch what you ask for and change how statistics are gathered based on that

```
SELECT histogram
FROM   user_tab_cols
WHERE  table_name = 'T'
AND    column_name = 'STATUS';

HISTOGRAM
---------------
NONE
```

```
SELECT histogram
FROM   user_tab_cols
WHERE  table_name = 'T'
AND    column_name = 'STATUS';

HISTOGRAM
---------------
FREQUENCY
```

You_Are_Being_Watched.sql

DBAces

Israel Oracle User Group

Oracle Database 12c – Adaptive Query Optimization

- Better SQL execution with intervention
- Optimizer is able to learn from its mistakes
- Set of capabilities that enable the optimizer to make run-time adjustments to execution plans and discover additional information that can lead to better statistics
- Two distinct aspects:
  - Adaptive plans
  - Adaptive statistics

DBAces

Israel Oracle User Group

Oracle Database 12c – Adaptive Plans

- Enables the optimizer to defer the final plan decision for a statement until execution time
- At runtime, optimizer can detect if its cardinality estimates differ greatly from the actual number of rows seen by the operations in the plan
- If there is a significant difference then the plan or a portion of it can be automatically adapted to avoid suboptimal performance on the first execution of a SQL statement

DBAces

Israel Oracle User Group

Oracle Database 12c – Enhanced Statistics

- Enhanced dynamic sampling
- Hybrid histograms
- Statistics during loads
- Session private statistics for GTT’s

DBAces

ilOUG
Israel Oracle User Group

Thank You !

DBAces

Ami Aharonovich

Oracle ACE & OCP

Ami@DBAces.co.il