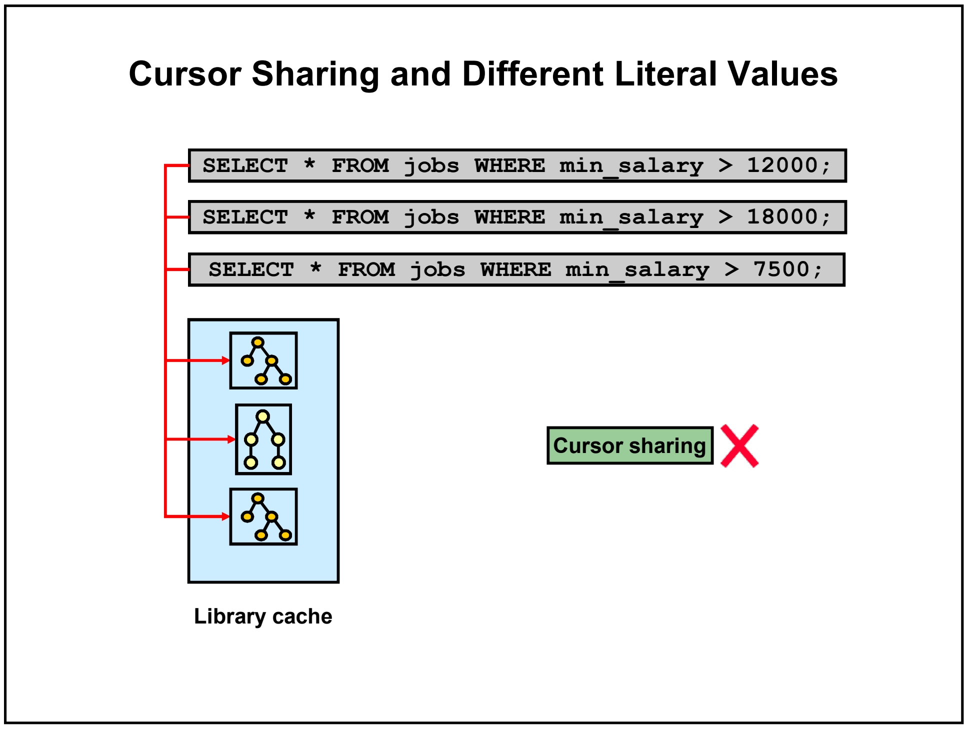
11 Using Bind Variables

Objectives

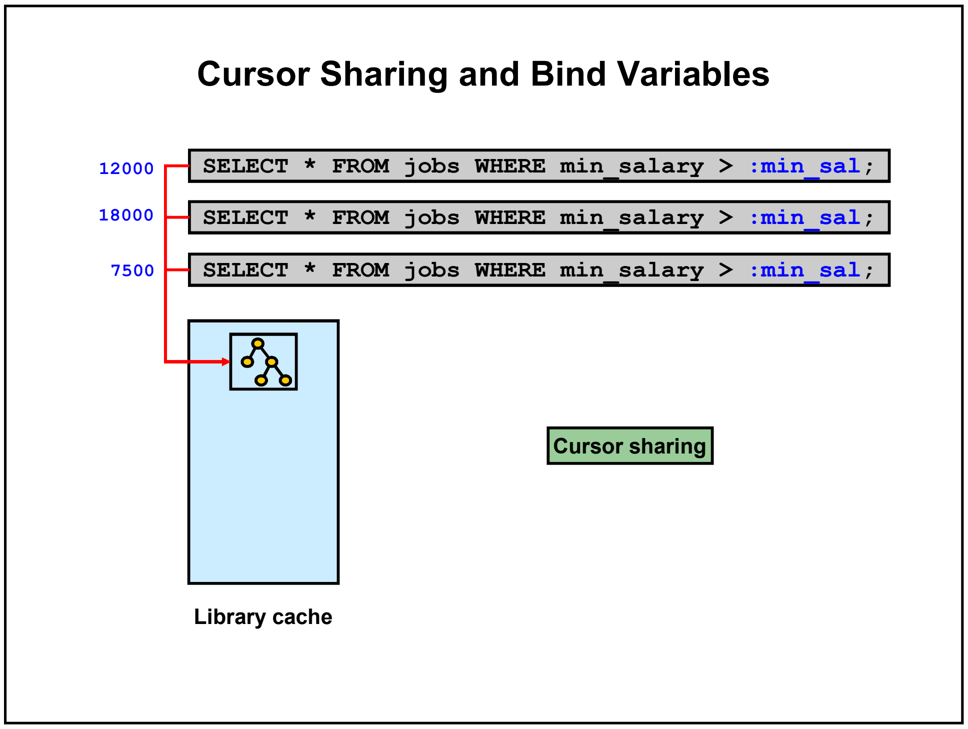
After completing this lesson, you should be able to:

- List the benefits of using bind variables
- Use bind peeking
- Use adaptive cursor sharing
- Describe common observations



Cursor Sharing and Different Literal Values

```
SELECT * FROM jobs WHERE min_salary > 12000;
SELECT * FROM jobs WHERE min_salary > 18000;
SELECT * FROM jobs WHERE min_salary > 7500;
```



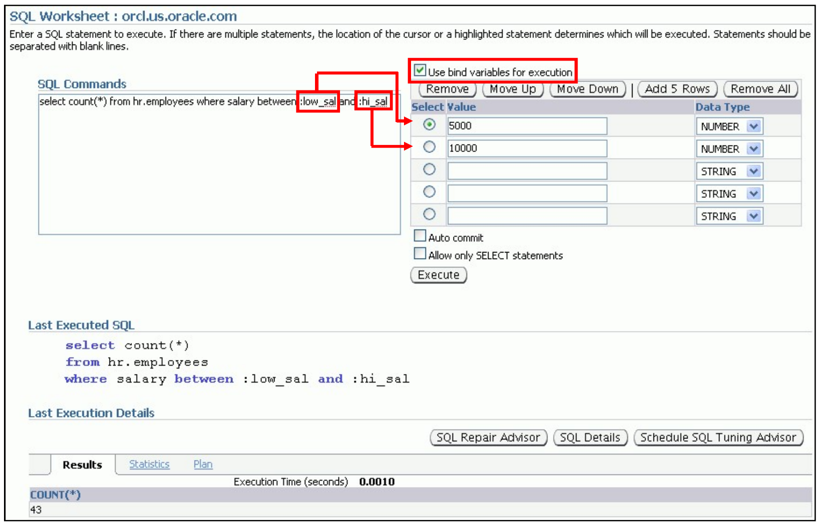
Cursor Sharing and Bind Variables

```
12000  SELECT * FROM jobs WHERE min_salary > :min_sal;
18000  SELECT * FROM jobs WHERE min_salary > :min_sal;
7500   SELECT * FROM jobs WHERE min_salary > :min_sal;
```

Bind Variables in SQL*Plus

```
SQL> variable job_id varchar2(10)
SQL> exec :job_id := 'SA_REP';

PL/SQL procedure successfully completed.

SQL> select count(*) from employees where job_id = :job_id;

  COUNT(*)
----------
        30

SQL> exec :job_id := 'AD_VP';

PL/SQL procedure successfully completed.

SQL> select count(*) from employees where job_id = :job_id;

  COUNT(*)
----------
         2
```

Bind Variables in Enterprise Manager



SQL Worksheet : orcl.us.oracle.com

Enter a SQL statement to execute. If there are multiple statements, the location of the cursor or a highlighted statement determines which will be executed. Statements should be separated with blank lines.

SQL Commands

```
select count(*) from hr.employees where salary between :low_sal and :hi_sal
```

[✓] Use bind variables for execution

Select	Value	Data Type
●	5000	NUMBER
○	10000	NUMBER
○		STRING
○		STRING
○		STRING

☐ Auto commit

☐ Allow only SELECT statements

Last Executed SQL

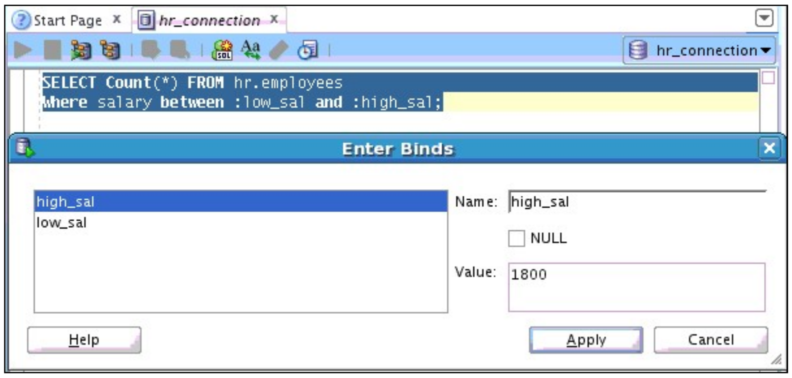
```
select count(*)
from hr.employees
where salary between :low_sal and :hi_sal
```

Last Execution Details

Execution Time (seconds) **0.0010**

```
COUNT(*)
43
```

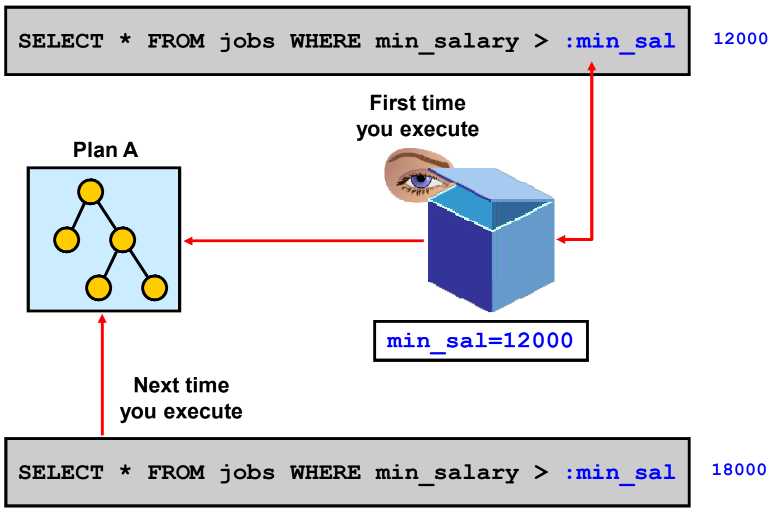
Bind Variables in Oracle SQL Developer



```
SELECT Count(*) FROM hr.employees
where salary between :low_sal and :high_sal;
```

Binds	high_sal, low_sal
Name:	high_sal
NULL	☐
Value:	1800

Bind Variable Peeking

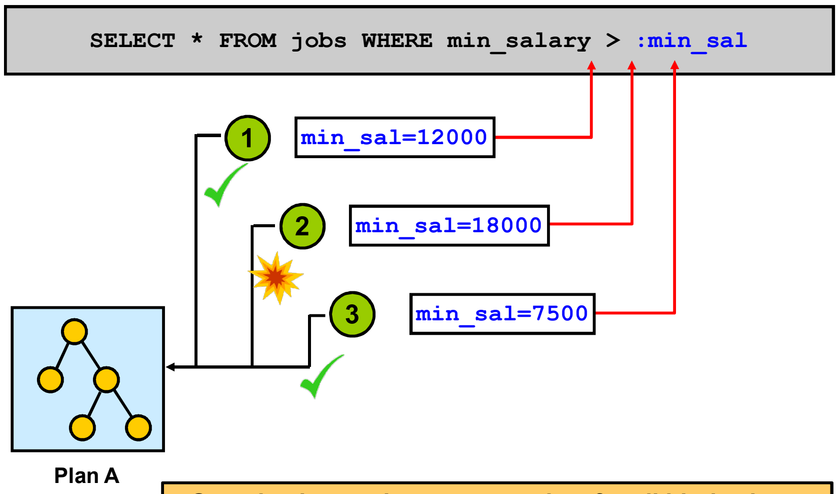


```
SELECT * FROM jobs WHERE min_salary > :min_sal     12000
```

```
min_sal=12000
```

```
SELECT * FROM jobs WHERE min_salary > :min_sal     18000
```

Bind Variable Peeking


```
SELECT * FROM jobs WHERE min_salary > :min_sal
```

One plan is not always appropriate for all bind values.

Cursor Sharing Enhancements

- Oracle8*i* introduced the possibility of sharing SQL statements that differ only in literal values.
- Oracle9*i* extends this feature by limiting it only to similar statements, instead of forcing it.
- Similar: Regardless of the literal value, same execution plan

```
SQL> SELECT * FROM employees
  2  WHERE employee_id = 153;
```

- Not similar: Possible different execution plans for different literal values.

```
SQL> SELECT * FROM employees
  2  WHERE department_id = 50;
```

The CURSOR_SHARING Parameter

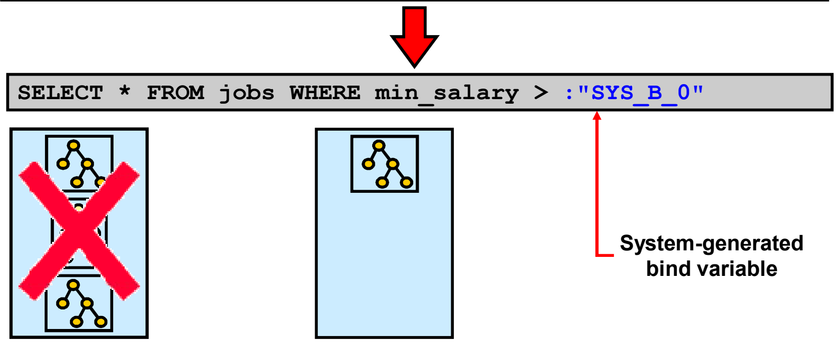
- CURSOR_SHARING parameter values:
  - FORCE
  - EXACT (default)
  - SIMILAR
- CURSOR_SHARING can be changed by using:
  - ALTER SYSTEM
  - ALTER SESSION
  - Initialization parameter files
- CURSOR_SHARING_EXACT hint

Forcing Cursor Sharing: Example

```
SQL> alter session set cursor_sharing = FORCE;
```

```
SELECT * FROM jobs WHERE min_salary > 12000;
SELECT * FROM jobs WHERE min_salary > 18000;
SELECT * FROM jobs WHERE min_salary > 7500;
```

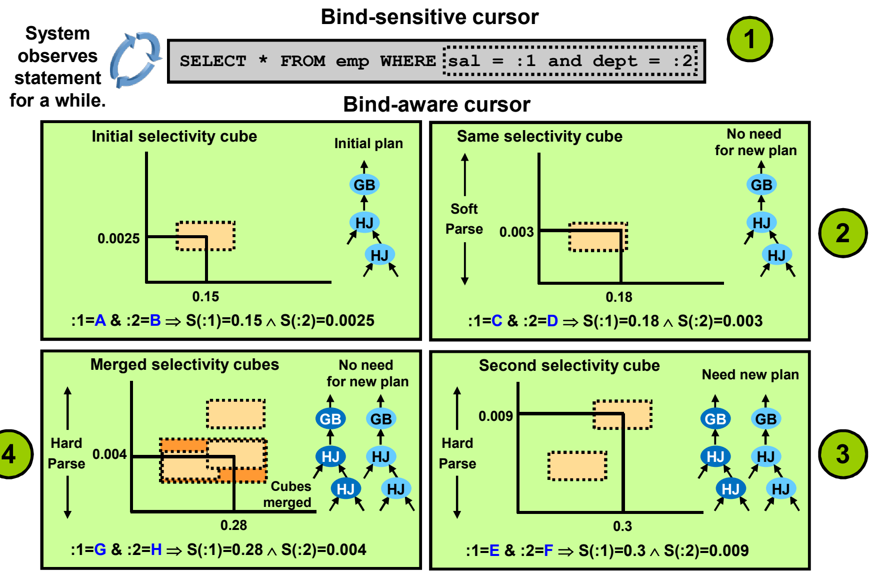
```
SELECT * FROM jobs WHERE min_salary > :"SYS_B_0"
```



Adaptive Cursor Sharing: Overview

- Allows for intelligent cursor sharing for statements that use bind variables
- Is used to compromise between cursor sharing and optimization
- Has the following benefits:
  - Automatically detects when different executions would benefit from different execution plans
  - Limits the number of generated child cursors to a minimum
  - Provides an automated mechanism that cannot be turned off

Adaptive Cursor Sharing: Architecture



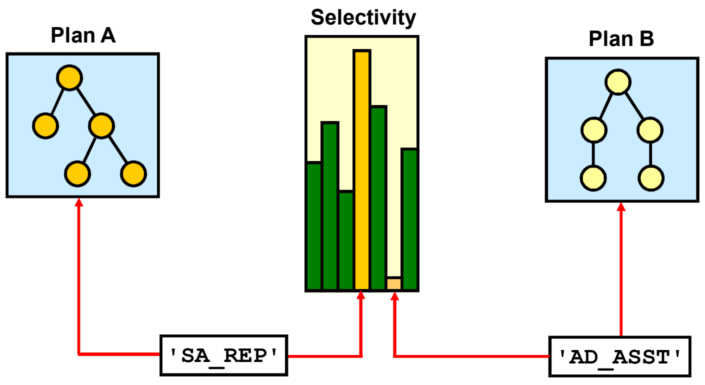
Adaptive Cursor Sharing: Views

The following views provide information about adaptive cursor sharing usage:

`V$SQL`	Two new columns show whether a cursor is bind sensitive or bind aware.
`V$SQL_CS_HISTOGRAM`	Shows the distribution of the execution count across the execution history histogram.
`V$SQL_CS_SELECTIVITY`	Shows the selectivity cubes stored for every predicate containing a bind variable and whose selectivity is used in the cursor sharing checks.
`V$SQL_CS_STATISTICS`	Shows execution statistics of a cursor using different bind sets.

Adaptive Cursor Sharing: Example

```
SQL> variable job varchar2(6)
SQL> exec :job := 'AD_ASST'
SQL> select count(*), max(salary) from employees where
    job_id=:job;
```



Interacting with Adaptive Cursor Sharing

- `CURSOR_SHARING`:
  - If `CURSOR_SHARING <> EXACT`, statements containing literals may be rewritten by using bind variables.
  - If statements are rewritten, adaptive cursor sharing may apply to them.
- SQL Plan Management (SPM):
  - If `OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES` is set to `TRUE`, only the first generated plan is used.
  - As a workaround, set this parameter to `FALSE`, and run your application until all plans are loaded in the cursor cache.
  - Manually load the cursor cache into the corresponding plan baseline.

Common Observations

Consider the following areas to resolve excessive parsing time as well:

- CPU time dominates the parse time
- Wait time dominates the parse time

```
SELECT * FROM ….

call     count       cpu    elapsed    disk     query   current     rows
------- ------  --------  --------  ------  --------  --------  -------
Parse      555    100.09    300.83       0         0         0        0
Execute    555      0.42      0.78       0         0         0        0
Fetch      555     14.04     85.03     513   1448514         0    11724
------- ------  --------  --------  ------  --------  --------  -------
total     1665    114.55    386.65     513   1448514         0    11724
```

Quiz

Which three statements are true about applications that are coded with literals rather than bind variables in the SQL statements?

a. More shared pool space is required for cursors.
b. Less shared pool space is required for cursors.
c. Histograms are used if available.
d. Histograms are not used.
e. No parsing is required for literal values.
f. Every different literal value requires parsing.

Quiz

The `CURSOR_SHARING` parameter should be set to _________ for systems with large tables and long-running queries, such as a data warehouse.

a. Similar
b. Force
c. Exact
d. Literal
e. True
f. False

Quiz

Adaptive cursor sharing can be turned off by setting the `CURSOR_SHARING` parameter to `FALSE`.

a. True

b. False

Summary

In this lesson, you should have learned how to:

- List the benefits of using bind variables
- Use bind peeking
- Use adaptive cursor sharing
- Describe common observations

Practice 11: Overview

This practice covers the following topics:

- Using adaptive cursor sharing and bind peeking
- Using the `CURSOR_SHARING` initialization parameter