8

Managing Data and Concurrency

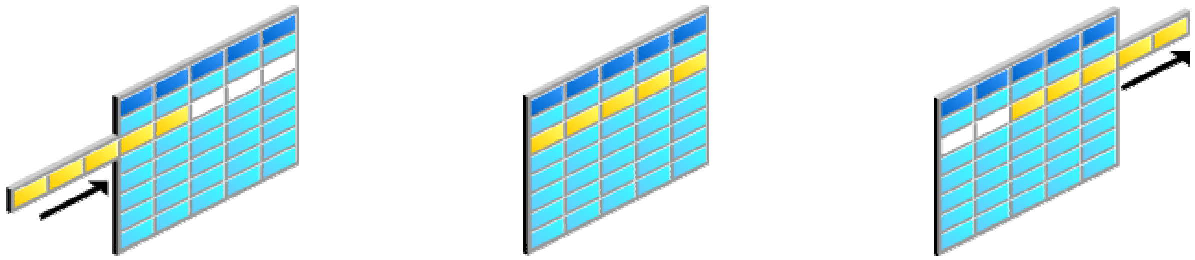
Objectives

After completing this lesson, you should be able to do the following:

- Manage data through the use of SQL
- Identify and administer PL/SQL objects
- Describe triggers and triggering events
- Monitor and resolve locking conflicts

Manipulating Data Through SQL

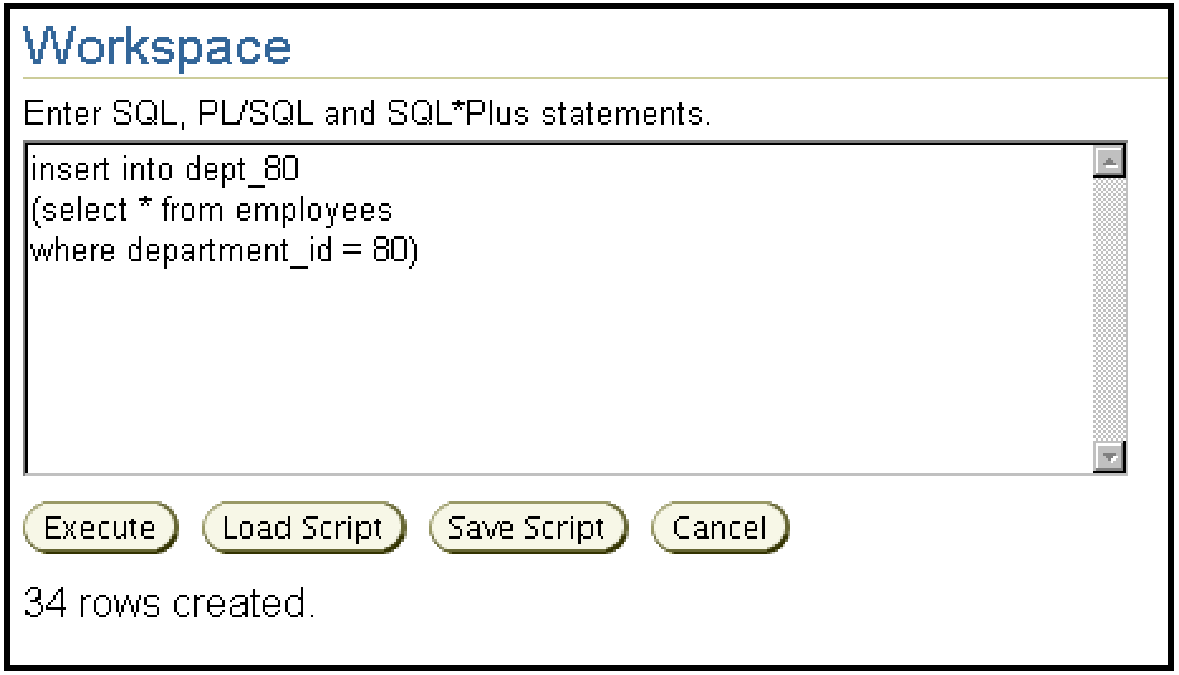
> **SQL**
> PL/SQL
> Locks



```
SQL> INSERT INTO employees VALUES
  2  (9999,'Bob','Builder','bob@abc.net',NULL,SYSDATE,
  3  'IT_PROG',NULL,NULL,100,90);

1 row created.

SQL> UPDATE employees SET SALARY=6000
  2  WHERE EMPLOYEE_ID = 9999;

1 row updated.

SQL> DELETE from employees
  2  WHERE EMPLOYEE_ID = 9999;

1 row deleted.
```

The `INSERT` Command

- Create one row at a time.
- Insert many rows from another table.

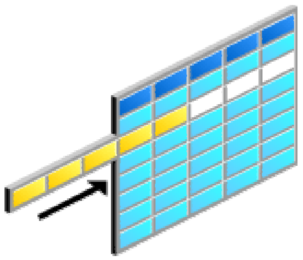


Workspace

Enter SQL, PL/SQL and SQL*Plus statements.

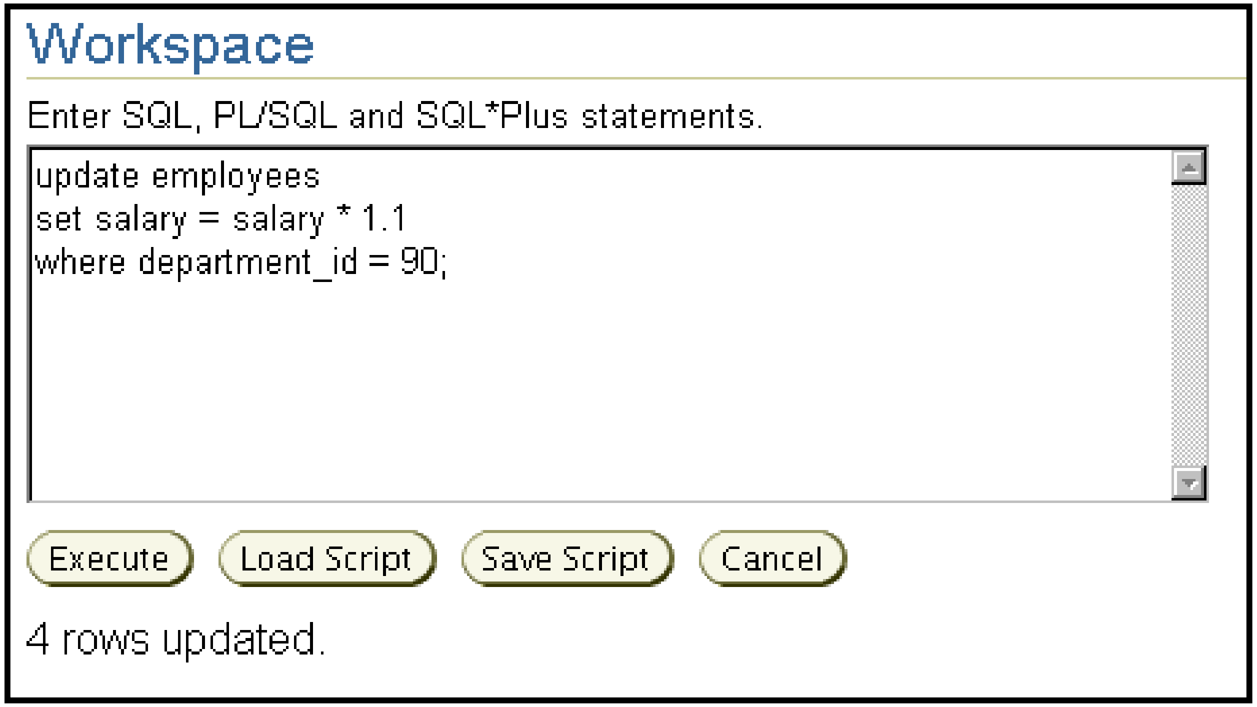
```
insert into dept_80
(select * from employees
where department_id = 80)
```

```
34 rows created.
```



The `UPDATE` Command

Use the `UPDATE` command to change zero or more rows of a table.

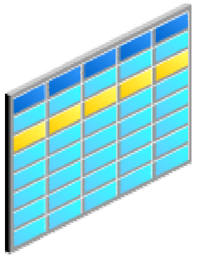


Workspace

Enter SQL, PL/SQL and SQL*Plus statements.

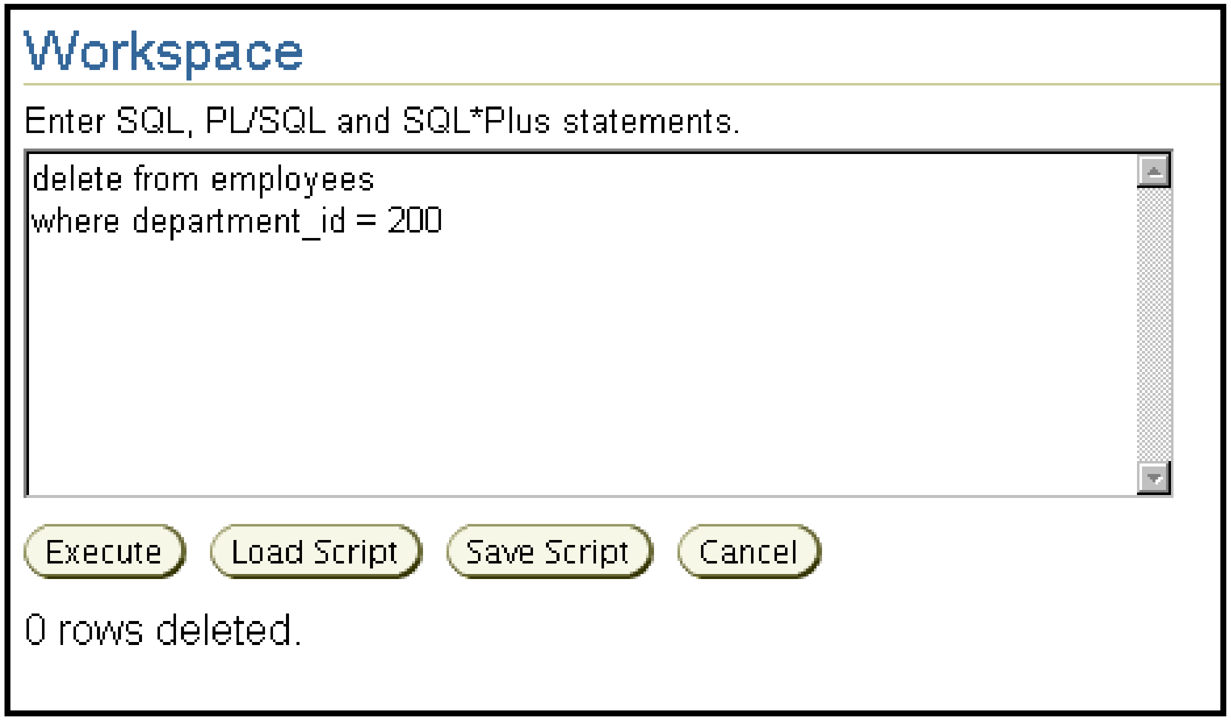
```
update employees
set salary = salary * 1.1
where department_id = 90;
```

```
4 rows updated.
```



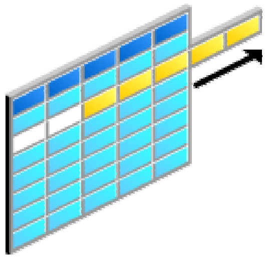
The DELETE Command

Use the `DELETE` command to remove zero or more rows from a table.



```
delete from employees
where department_id = 200
```

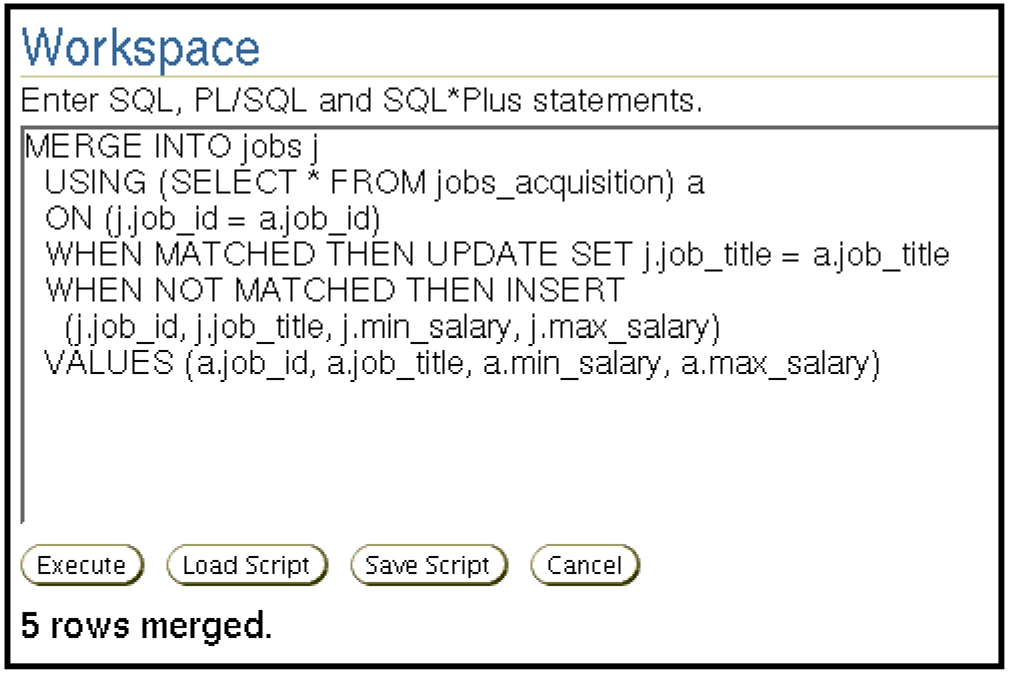
```
0 rows deleted.
```



ORACLE

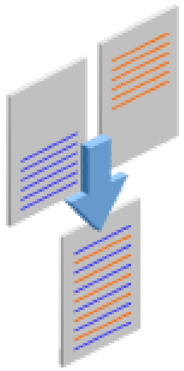
The MERGE Command

Use the `MERGE` command to perform both `INSERT` and `UPDATE` in a single command.



```
MERGE INTO jobs j
 USING (SELECT * FROM jobs_acquisition) a
 ON (j.job_id = a.job_id)
 WHEN MATCHED THEN UPDATE SET j.job_title = a.job_title
 WHEN NOT MATCHED THEN INSERT
  (j.job_id, j.job_title, j.min_salary, j.max_salary)
 VALUES (a.job_id, a.job_title, a.min_salary, a.max_salary)
```

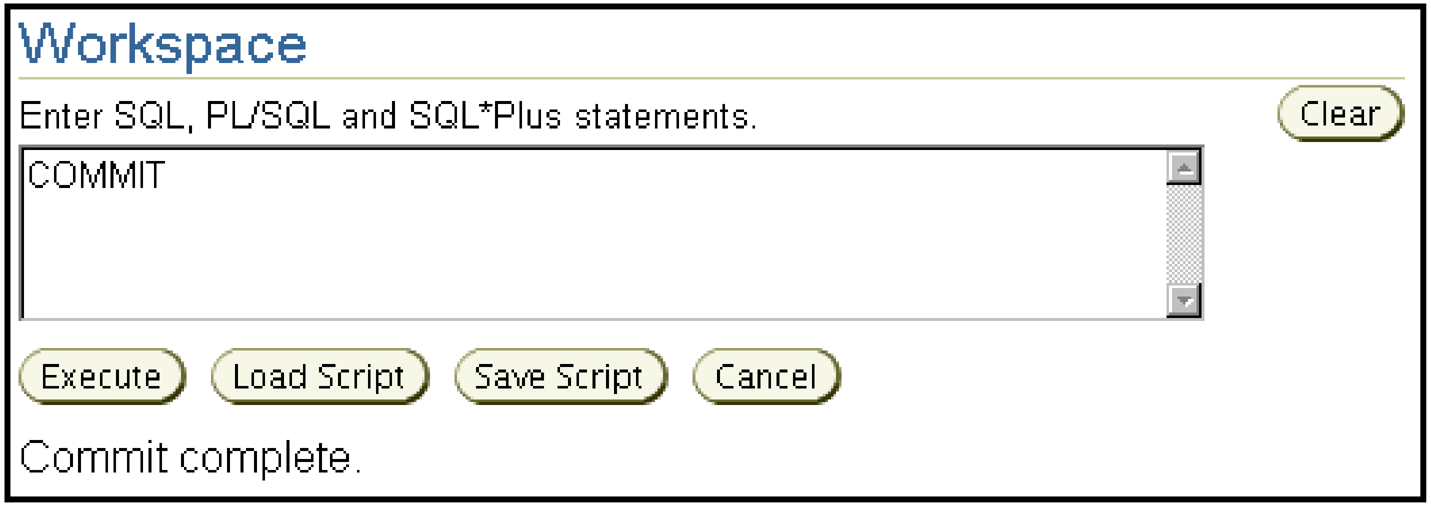
```
5 rows merged.
```



The `COMMIT` and `ROLLBACK` Commands

The following are used to finish a transaction:

- `COMMIT`: **Makes the change permanent**
- `ROLLBACK`: **Undoes the change**



```
COMMIT
```

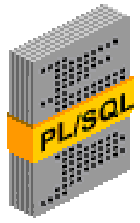
```
Commit complete.
```

PL/SQL

SQL
> PL/SQL
Locks

Oracle's Procedural Language extension to SQL (PL/SQL) is a fourth-generation programming language (4GL). It provides:

- Procedural extensions to SQL
- Portability across platforms and products
- Higher level of security and data integrity protection
- Support for object-oriented programming

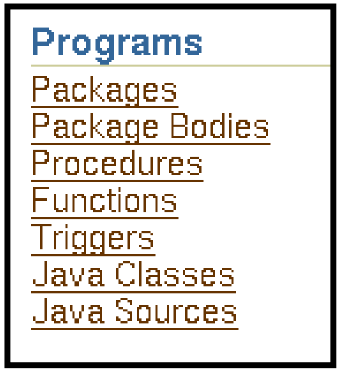


ORACLE

Administering PL/SQL Objects

Database administrators should be able to:

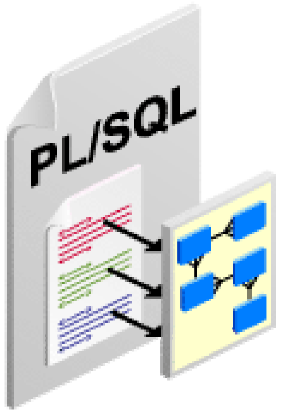
- Identify problem PL/SQL objects
- Recommend the appropriate use of PL/SQL
- Load PL/SQL objects into the database
- Assist PL/SQL developers in troubleshooting


Programs
Packages
Package Bodies
Procedures
Functions
Triggers
Java Classes
Java Sources

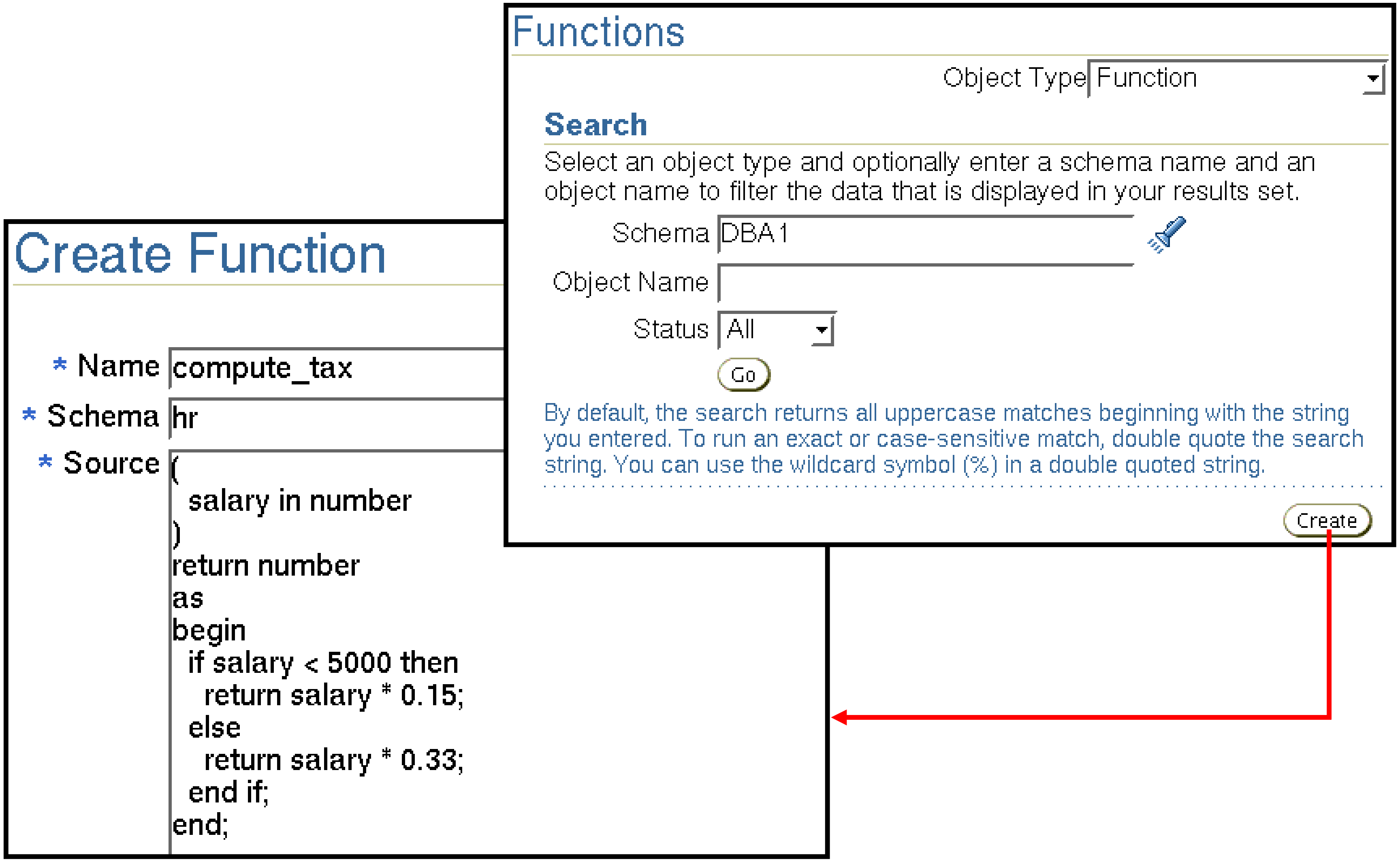
PL/SQL Objects

There are many types of PL/SQL database objects:

- Package
- Package body
- Type body
- Procedure
- Function
- Trigger



Functions



Functions

Object Type: Function

Search

Select an object type and optionally enter a schema name and an object name to filter the data that is displayed in your results set.

Schema: DBA1

Object Name:

Status: All

By default, the search returns all uppercase matches beginning with the string you entered. To run an exact or case-sensitive match, double quote the search string. You can use the wildcard symbol (%) in a double quoted string.

Create Function

* Name: compute_tax

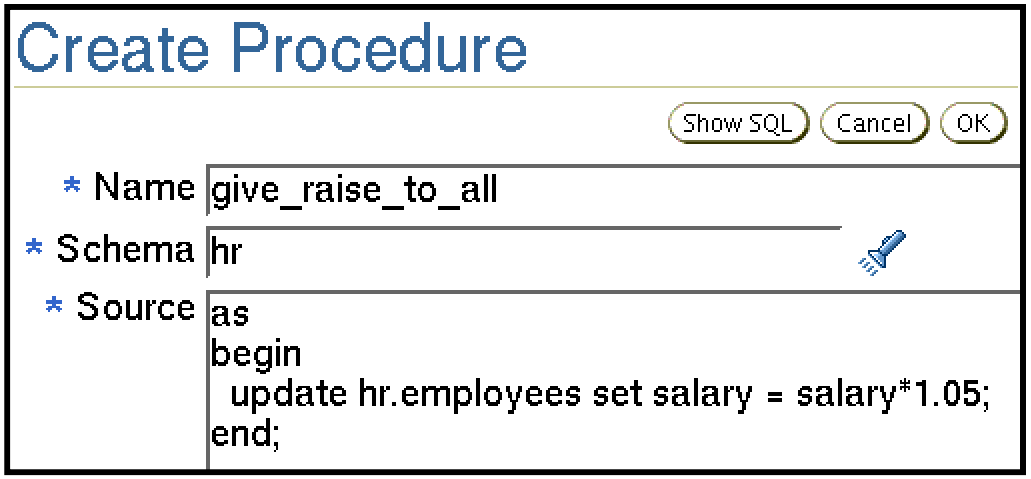
* Schema: hr

* Source:

```
(
  salary in number
)
return number
as
begin
  if salary < 5000 then
    return salary * 0.15;
  else
    return salary * 0.33;
  end if;
end;
```

Procedures

- Are used to perform a specific action
- Pass values in and out by using an argument list
- Can be invoked using:
  - The `CALL` command, which is a SQL statement
  - The `EXECUTE` command, which is a SQL*Plus command



* Name	give_raise_to_all
* Schema	hr

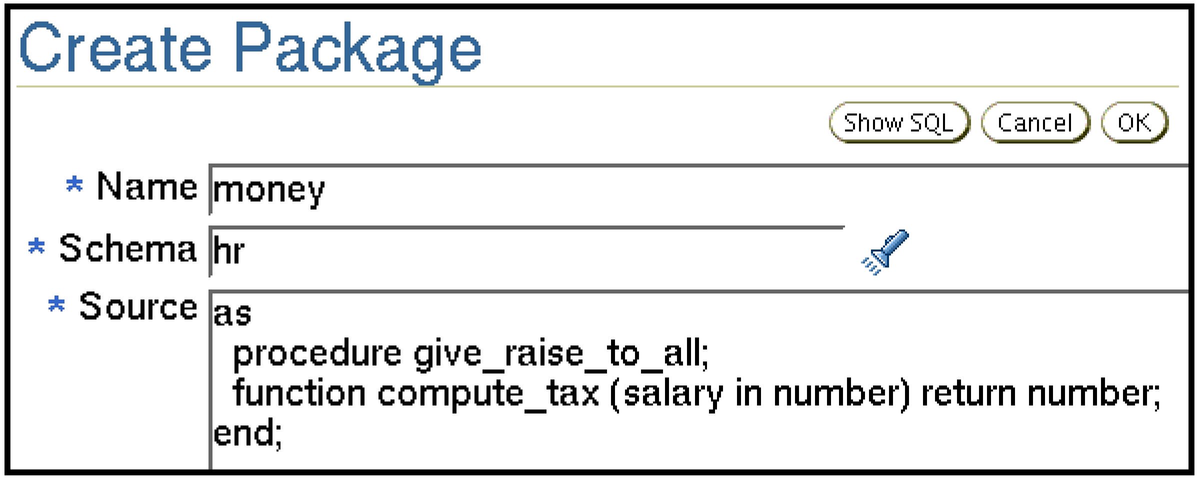
* Source

```
as
begin
  update hr.employees set salary = salary*1.05;
end;
```

Packages

Packages are collections of functions and procedures. Each package should consist of two objects:

- Package specification
- Package body



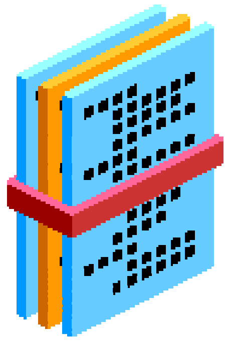
* Name: money

* Schema: hr

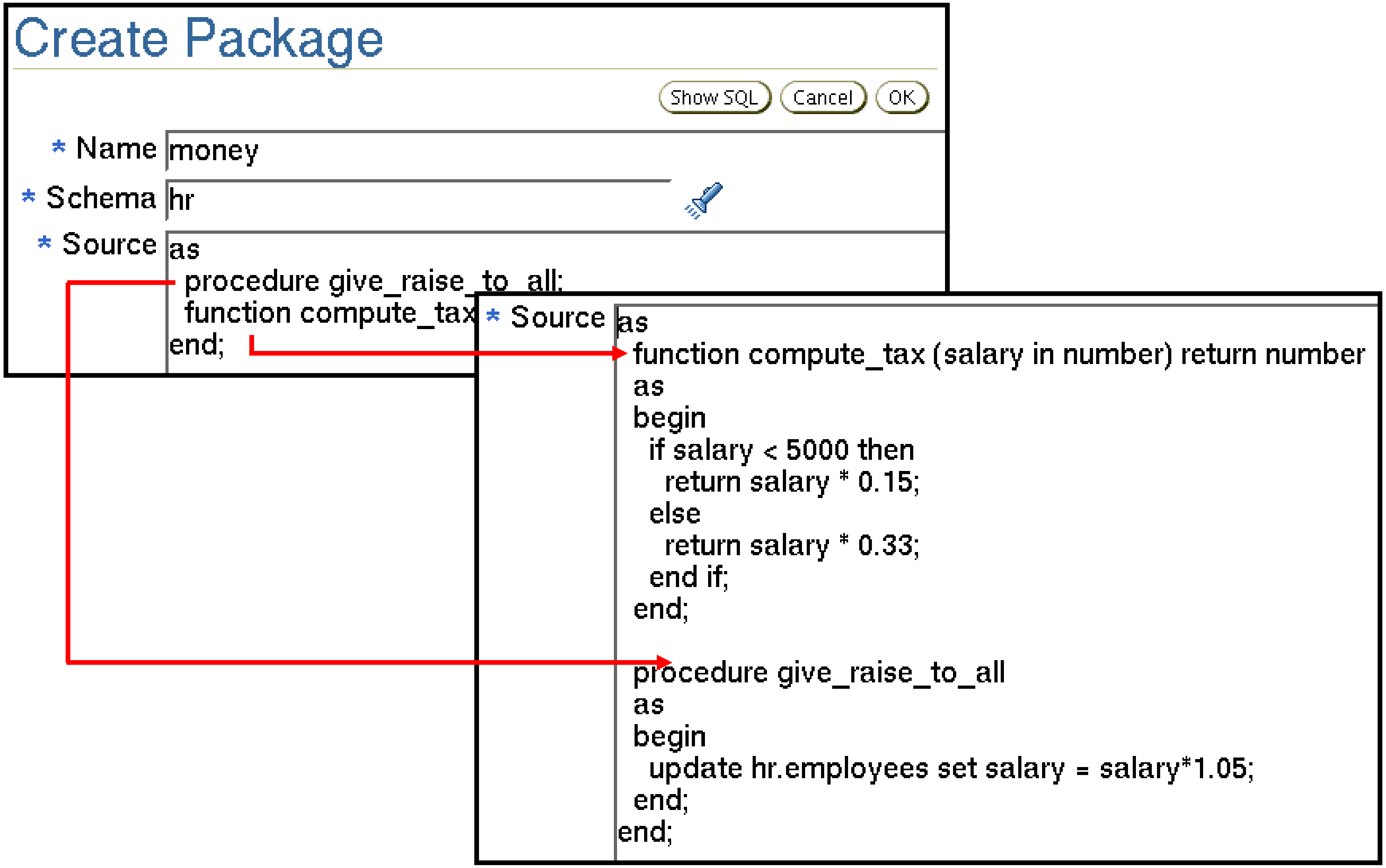
* Source:

```
as
  procedure give_raise_to_all;
  function compute_tax (salary in number) return number;
end;
```

Package specification



Package Specification and Body



Create Package

Show SQL | Cancel | OK

* Name: money

* Schema: hr

* Source:

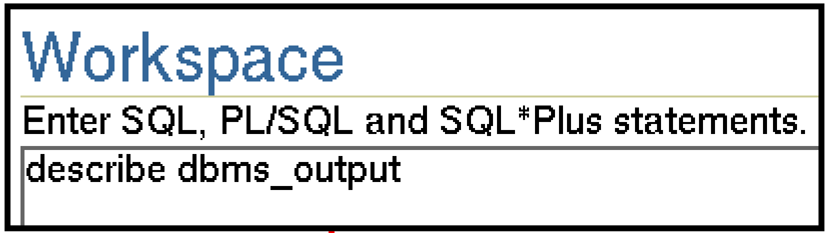
```
as
  procedure give_raise_to_all;
  function compute_tax
end;
```

* Source:

```
as
 function compute_tax (salary in number) return number
 as
 begin
   if salary < 5000 then
     return salary * 0.15;
   else
     return salary * 0.33;
   end if;
 end;

 procedure give_raise_to_all
 as
 begin
   update hr.employees set salary = salary*1.05;
 end;
end;
```

Built-in Packages

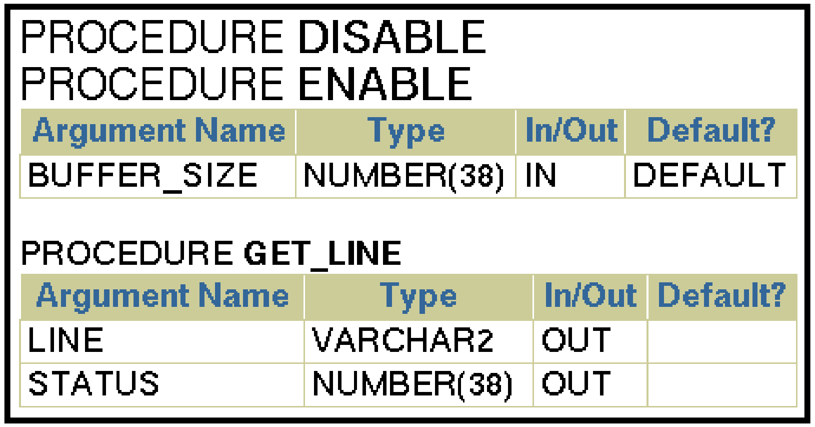
- The Oracle database comes with over 350 built-in PL/SQL packages, which provide:
  - Administration and maintenance utilities
  - Extended functionality
- Use the `DESCRIBE` command to view subprograms.



Workspace

Enter SQL, PL/SQL and SQL*Plus statements.

```
describe dbms_output
```



PROCEDURE DISABLE

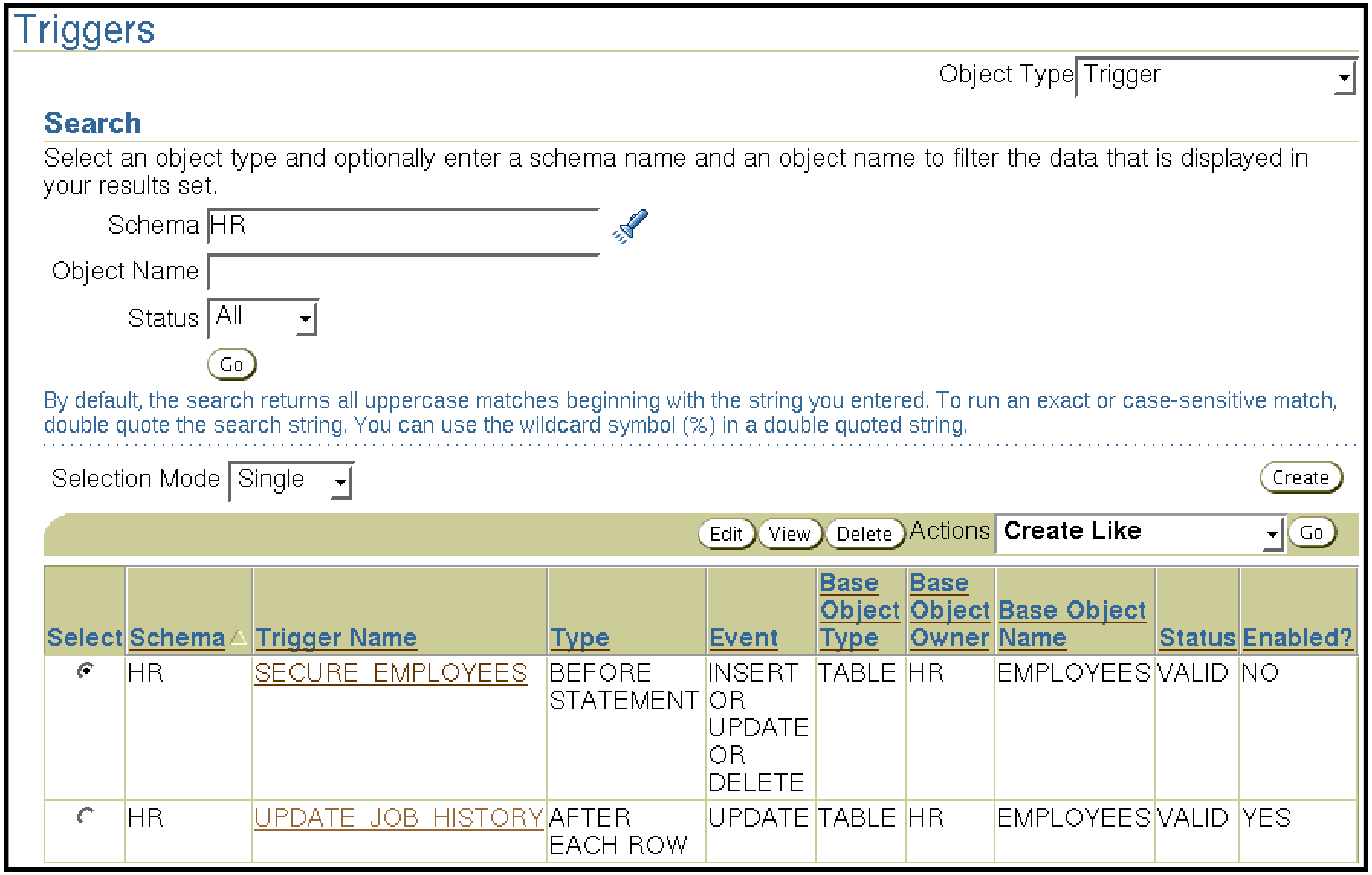
PROCEDURE ENABLE

Argument Name	Type	In/Out	Default?
BUFFER_SIZE	NUMBER(38)	IN	DEFAULT

PROCEDURE GET_LINE

Argument Name	Type	In/Out	Default?
LINE	VARCHAR2	OUT	
STATUS	NUMBER(38)	OUT	

Triggers



Triggers

Object Type: Trigger

Search

Select an object type and optionally enter a schema name and an object name to filter the data that is displayed in your results set.

Schema: HR

Object Name:

Status: All

Go

By default, the search returns all uppercase matches beginning with the string you entered. To run an exact or case-sensitive match, double quote the search string. You can use the wildcard symbol (%) in a double quoted string.

Selection Mode: Single

Create

Edit | View | Delete | Actions: Create Like | Go

Select	Schema	Trigger Name	Type	Event	Base Object Type	Base Object Owner	Base Object Name	Status	Enabled?
●	HR	SECURE_EMPLOYEES	BEFORE STATEMENT	INSERT OR UPDATE OR DELETE	TABLE	HR	EMPLOYEES	VALID	NO
○	HR	UPDATE_JOB_HISTORY	AFTER EACH ROW	UPDATE	TABLE	HR	EMPLOYEES	VALID	YES

Triggering Events

Event Type	Examples of Events
DML	`INSERT, UPDATE, DELETE`
DDL	`CREATE, DROP, ALTER, GRANT, REVOKE, RENAME`
Database	`LOGON, LOGOFF, STARTUP, SHUTDOWN, SERVERERROR, SUSPEND`

Locks

- Locks prevent multiple sessions from changing the same data at the same time.
- They are automatically obtained at the lowest possible level for a given statement.
- They do not escalate.

Transaction 1

Transaction 2

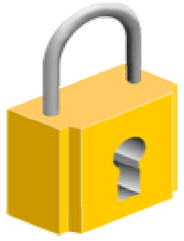
```
SQL> UPDATE employees
  2   SET salary=salary+100
  3   WHERE employee_id=100;
```

```
SQL> UPDATE employees
  2   SET salary=salary*1.1
  3 WHERE employee_id=100;
```

Locking Mechanism

- High level of data concurrency:
  - Row-level locks for inserts, updates, and deletes
  - No locks required for queries
- Automatic queue management
- Locks held until the transaction ends (with the `COMMIT` or `ROLLBACK` operation)

Transaction 1



Transaction 2 ✓

```
SQL> UPDATE employees
  2  SET salary=salary+100
  3  WHERE employee_id=100;
```

```
SQL> UPDATE employees
  2  SET salary=salary*1.1
  3  WHERE employee_id=101;
```

Data Concurrency

Time:	**Transaction 1**	`UPDATE hr.employees` `SET salary=salary+100` `WHERE employee_id=100;`
	Transaction 2	`UPDATE hr.employees` `SET salary=salary+100` `WHERE employee_id=101;`
09:00:00	**Transaction 3**	`UPDATE hr.employees` `SET salary=salary+100` `WHERE employee_id=102;`
	...	**...**
	Transaction *x*	`UPDATE hr.employees` `SET salary=salary+100` `WHERE employee_id=`***`xxx`***`;`

DML Locks

Transaction 1

```
SQL> UPDATE employees
  2  SET salary=salary*1.1
  3  WHERE employee_id= 107;
1 row updated.
```

Transaction 2

```
SQL> UPDATE employees
  2  SET salary=salary*1.1
  3  WHERE employee_id= 106;
1 row updated.
```

Each DML transaction must acquire *two* locks:

- `EXCLUSIVE` **row lock for the row or rows being updated**
- `ROW EXCLUSIVE` **table-level lock for the table containing the rows**

Enqueue Mechanism

The enqueue mechanism keeps track of:

- Sessions waiting for locks
- The requested lock mode
- The order in which sessions requested the lock

Lock Conflicts

Transaction 1	Time	Transaction 2

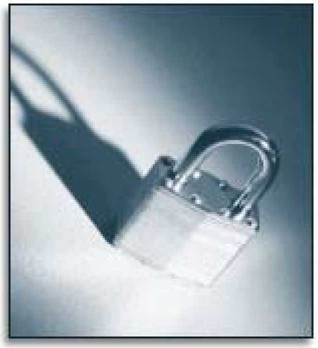
`UPDATE employees SET salary=salary+100 WHERE employee_id=100;`
`1 row updated.`
	9:00:00	`UPDATE employees SET salary=salary+100 WHERE employee_id=101;`
		`1 row updated.`

`UPDATE employees SET COMMISION_PCT=2 WHERE employee_id=101;`
Session waits enqueued due to lock conflict.
	9:00:05	`SELECT sum(salary) FROM employees;`
		`SUM(SALARY)`
		`-----------`
		`     692634`

Session still waiting!	16:30:00	**Many selects, inserts, updates, and deletes during the last 7.5 hours, but no commits or rollbacks!**
`1 row updated.` **Session continues.**	16:30:01	`commit;`

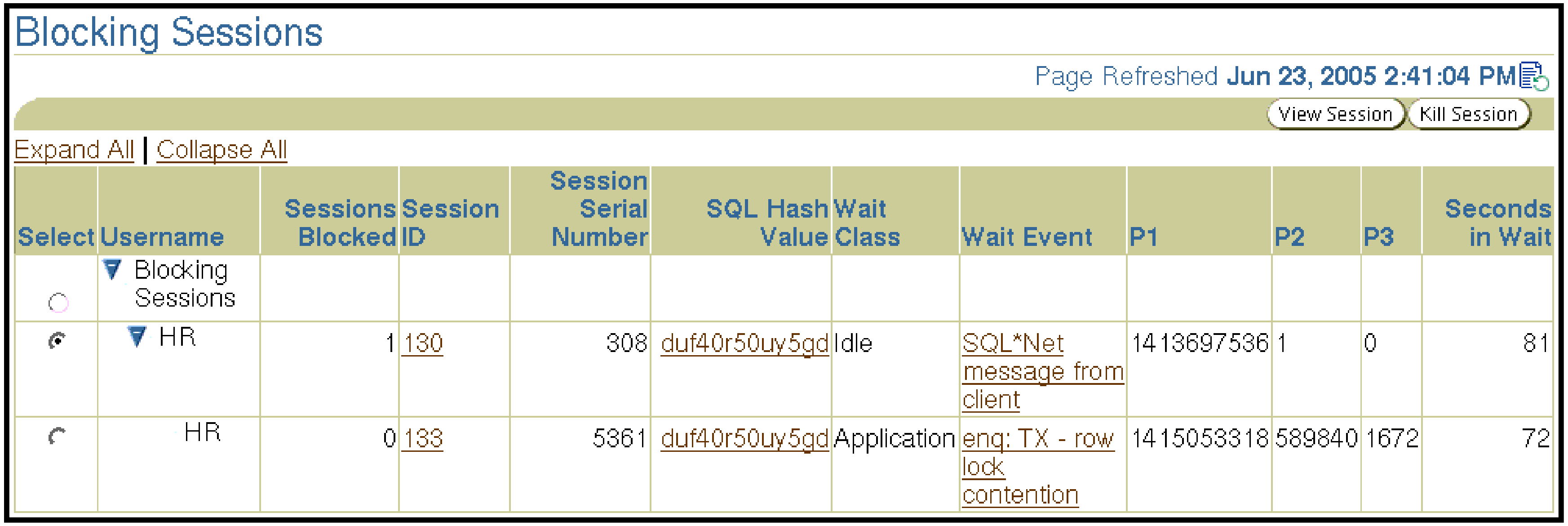
Possible Causes of Lock Conflicts

- Uncommitted changes
- Long-running transactions
- Unnecessarily high locking levels



Detecting Lock Conflicts

Select Blocking Sessions from the Performance page.



Blocking Sessions

Page Refreshed **Jun 23, 2005 2:41:04 PM**

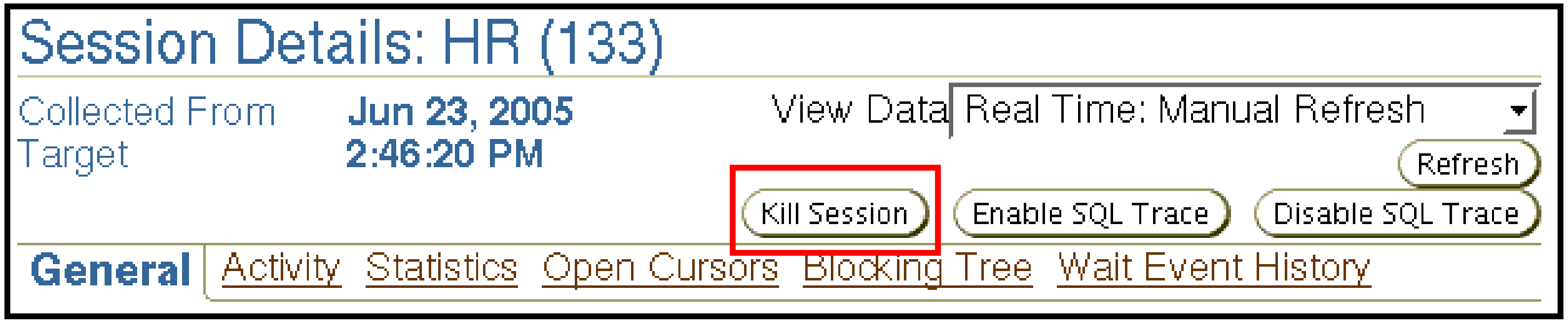
Select	Username	Sessions Blocked	Session ID	Session Serial Number	SQL Hash Value	Wait Class	Wait Event	P1	P2	P3	Seconds in Wait
○	▼ Blocking Sessions										
●	▼ HR	1	130	308	duf40r50uy5gd	Idle	SQL*Net message from client	1413697536	1	0	81
○	HR	0	133	5361	duf40r50uy5gd	Application	enq: TX - row lock contention	1415053318	589840	1672	72

Click the Session ID link to view information about the locking session, including the actual SQL statement.

Resolving Lock Conflicts

To resolve a lock conflict:

- Have the session holding the lock commit or roll back
- Terminate the session holding the lock as a last resort

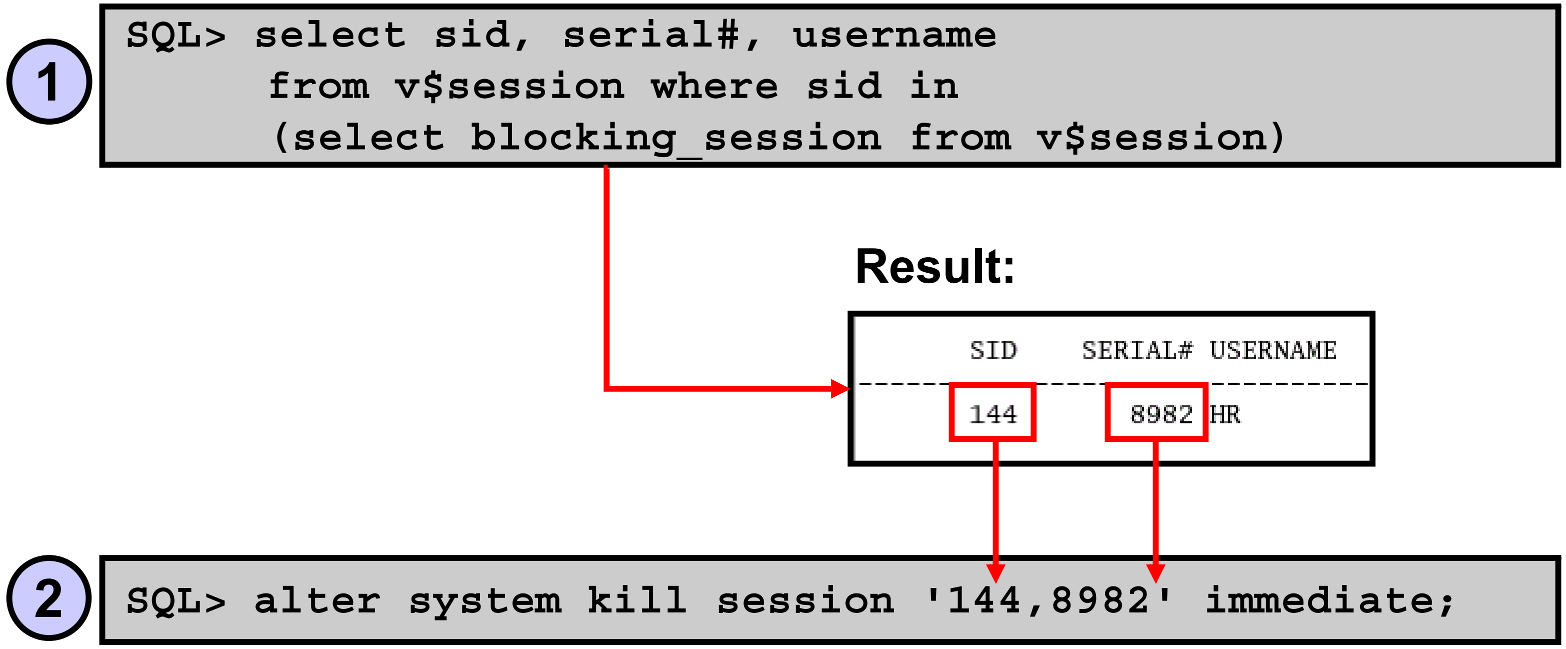

Session Details: HR (133)

Collected From Target: Jun 23, 2005 2:46:20 PM

View Data: Real Time: Manual Refresh

Resolving Lock Conflicts Using SQL

SQL statements can be used to determine the blocking session and kill it.



1

```
SQL> select sid, serial#, username
      from v$session where sid in
      (select blocking_session from v$session)
```

Result:

```
       SID    SERIAL# USERNAME
---------- ---------- ----------
       144       8982 HR
```

2

```
SQL> alter system kill session '144,8982' immediate;
```

Deadlocks

Transaction 1		Transaction 2

`UPDATE employees`
`SET salary = salary x 1.1`
`WHERE employee_id = 1000;`
	9:00	`UPDATE employees`
		`SET manager  = 1342`
		`WHERE employee_id = 2000;`

`UPDATE employees`
`SET salary = salary x 1.1`
`WHERE employee_id = 2000;`
	9:15	`UPDATE employees`
		`SET manager  = 1342`
		`WHERE employee_id = 1000;`

`ORA-00060:`
`Deadlock detected while`
`waiting for resource`
	9:16	

Summary

In this lesson, you should have learned how to:

- Manage data through the use of SQL
- Identify and administer PL/SQL objects
- Describe triggers and triggering events
- Monitor and resolve locking conflicts

Practice Overview: Managing Data and Concurrency

This practice covers the following topics:

- Identifying locking conflicts
- Resolving locking conflicts