
ORACLE
ACADEMY

Database Programming with PL/SQL

9-6
Using Invoker's Rights and Autonomous Transactions

ORACLE ACADEMY

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Objectives

This lesson covers the following objectives:

- Contrast invoker's rights with definer's rights
- Create a procedure that uses invoker's rights
- Create a procedure that executes an Autonomous Transaction

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Purpose

- In the previous lesson, you learned that the creator (owner) of a PL/SQL subprogram must be granted the relevant object privileges on the tables that are referenced by SQL statements within the subprogram.
- Also, table names, view names, and so on are assumed to be in the subprogram creator's schema unless explicitly prefixed with a different schema-name.
- This way of checking object references is called Definer's Rights.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Purpose

- What if you want the invoking user's table names, etc. to be used instead?
- You must understand and use Invoker's Rights.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Using Definer's Rights

- "Definer" means the owner (usually the creator) of the PL/SQL subprogram.
- "Invoker" means the user who calls (invokes) the subprogram.
- When Definer's Rights are used (the default):
  - The definer needs privileges on the database objects referenced within the subprogram.
  - The invoker only needs EXECUTE privilege on the subprogram.
- The subprogram executes in the user's database session, but with the privileges and table names of the definer.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Using Definer's Rights: Example 1

- Both Tom and Sue own a TESTS table.
- Sue executes Tom's procedure.
- Whose TESTS table is used?
- Does Sue need SELECT privilege on TOM.TESTS?
- Why or why not?

```
Tom> CREATE TABLE tests ... ;
Sue> CREATE TABLE tests ... ;

Tom> CREATE OR REPLACE PROCEDURE grades ...
     IS BEGIN
       ... SELECT ... FROM tests ... ;      END;

Tom> GRANT EXECUTE ON grades TO sue;

Sue> BEGIN ... tom.grades(...); ... END;
```


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Using Definer's Rights: Example 2

- When using Definer's Rights, table-name TESTS is resolved in the definer's schema.
- This means that table TESTS is assumed to be in Bill's schema, so you do not need to prefix the table-name with the schema-name, or create a synonym.
- This procedure compiles and executes successfully provided the relevant privileges have been granted.

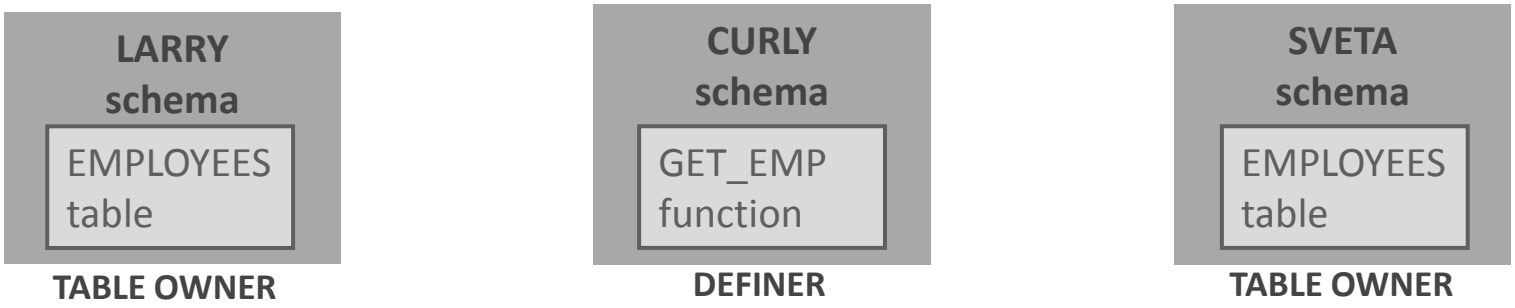
```
Bill>  CREATE TABLE tests ... ;
Bill>  CREATE OR REPLACE PROCEDURE grades ... IS
       BEGIN
       ... SELECT ... FROM tests ... ;
       END;

Eylem> BEGIN ... bill.grades(...); END;
```


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Using Definer's Rights: Example 3

- Which EMPLOYEES table will LARRY select from?
- What if Sveta (or Curly) executes the function?




```
Larry> GRANT SELECT ON employees TO curly;
Sveta> GRANT SELECT ON employees TO curly;

Curly> CREATE SYNONYM employees FOR sveta.employees;
Curly> CREATE FUNCTION get_emp ... BEGIN ...
         SELECT ... FROM employees; ... END;
Curly> GRANT EXECUTE ON get_emp TO larry, sveta;

Larry> ... BEGIN ... v_result := curly.get_emp; END;
```


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Using Invoker's Rights

- When Invoker's Rights are used:
  - The invoker needs privileges on the database objects referenced within the subprogram, as well as EXECUTE privilege on the procedure
  - The definer does not need any privileges.
- The subprogram executes in the user's database session, with the user's privileges, table references, and synonyms.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Using Invoker's Rights

- Set AUTHID to CURRENT_USER, immediately before IS | AS:

```
Tom> CREATE OR REPLACE FUNCTION grades
      (p_name IN VARCHAR2) RETURN NUMBER
       AUTHID CURRENT_USER IS
       v_score  NUMBER;
     BEGIN
       SELECT score INTO v_score FROM tests
         WHERE key=p_name;
       RETURN v_score;
     END;
```

- Now the user's privileges and table references (including synonyms) are checked at execution time.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Using Invoker's Rights: Example 1 Revisited

- Which TESTS table will Sue select from now?
- And which privileges does Sue need?

```
Tom> CREATE TABLE tests ... ;
Sue> CREATE TABLE tests ... ;

Tom> CREATE OR REPLACE PROCEDURE grades ...
     AUTHID CURRENT_USER IS BEGIN
       ... SELECT ... FROM tests ... ;
     END;
Tom> GRANT EXECUTE ON grades TO sue;

Sue> BEGIN ... tom.grades(...); ... END;
```


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Using Invoker's Rights: Example 2 Revisited

- Bill changes the procedure to use Invoker's Rights:

```
Bill>  CREATE TABLE tests ... ;
Bill>  CREATE OR REPLACE PROCEDURE grades ...
       AUTHID CURRENT_USER IS
       BEGIN
       ... SELECT ... FROM tests ... ;
       END;
Eylem> BEGIN ... bill.grades(...); END;
```

- Using Invoker's Rights, object-names are resolved in the invoker's schema.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

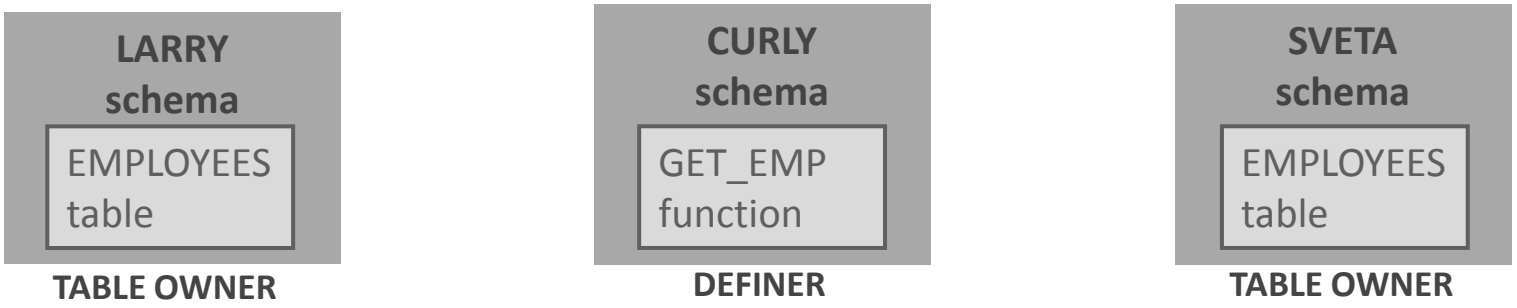
Using Invoker's Rights: Example 2 Revisited

- This means that TESTS is now assumed to be in Eylem's schema.
- But Eylem does not own a TESTS table.
- How could Bill change the procedure code to allow Eylem to execute the procedure successfully?

```
Bill>  CREATE TABLE tests ... ;
Bill>  CREATE OR REPLACE PROCEDURE grades ...
       AUTHID CURRENT_USER IS
       BEGIN
       ... SELECT ... FROM tests ... ;
       END;
Eylem> BEGIN ... bill.grades(...); END;
```


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Using Invoker's Rights: Example 3 Revisited




```
Larry> GRANT SELECT ON employees TO curly;
Sveta> GRANT SELECT ON employees TO curly;

Curly> CREATE SYNONYM employees FOR sveta.employees;
Curly> CREATE FUNCTION get_emp ... AUTHID CURRENT_USER IS
       BEGIN...SELECT...FROM employees;...END;
Curly> GRANT EXECUTE ON get_emp TO larry, sveta;

Larry> ... BEGIN ... v_result := curly.get_emp; END;
```

Now which EMPLOYEES table will LARRY select from?

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

A Final Example of Invoker's Rights:

What extra privilege and/or synonym does Hakan need in order to execute PAUL.ADD_DEPT successfully?

```
Paul> CREATE TABLE departments ... ;
Paul> CREATE OR REPLACE PROCEDURE add_dept ...
      AUTHID CURRENT_USER IS
      BEGIN
        INSERT INTO departments ...;
      END;
Paul> GRANT EXECUTE ON add_dept TO hakan;

Hakan> BEGIN ... paul.add_dept(...); END;
```


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Autonomous Transactions

- True or False: A single connected session can have only one active transaction at a time.
- Until you finish this transaction (by `COMMIT` or `ROLLBACK`) you cannot start another transaction.
- False!
- Using an Autonomous Transaction, your session can have two or more active transactions at the same time, which can `COMMIT` or `ROLLBACK` independently of each other.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

When is an Autonomous Transaction Needed?

- Imagine you have a bank account with an ATM card which allows you to withdraw cash from your bank account at a cash dispensing machine.
- Someone steals your ATM card and tries to use it illegally to withdraw your money.
- The attempt fails because the thief doesn't know your PIN number.
- Even though the cash withdrawal transaction is rolled back (and your money is safe), you would still want the bank to log (and commit) the failed attempt, so they can find and catch the thief.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Creating an Autonomous Transaction

- The autonomous transaction (AT) is called from within the main transaction (MT).
- The AT must be in a separate subprogram, with PRAGMA AUTONOMOUS_TRANSACTION; coded immediately after IS/AS.

```
PROCEDURE mt_proc IS
  emp_id  NUMBER;
BEGIN
  emp_id := 1234;
  INSERT ...
  at_proc;
  DELETE ...
  COMMIT; -- or ROLLBACK;
END mt_proc;
```

```
PROCEDURE at_proc IS
 PRAGMA
    AUTONOMOUS_TRANSACTION;
 dept_id  NUMBER := 90;
 BEGIN
   UPDATE ...
   INSERT ...
   COMMIT;   -- Required
 END at_proc;
```


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Creating an Autonomous Transaction

- Notice that `PRAGMA AUTONOMOUS_TRANSACTION;` must end with a semicolon (;), and comes after `IS/AS`.
- The Autonomous Transaction must be committed or rolled back within the same procedure.

```
PROCEDURE at_proc IS
 PRAGMA AUTONOMOUS_TRANSACTION;
 dept_id  NUMBER := 90;
 BEGIN
   UPDATE ... INSERT ...
   COMMIT;   -- (or ROLLBACK;) Required!
 END at_proc;
```


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Autonomous Transaction: ATM Card Example

```
PROCEDURE log_usage (p_card_id NUMBER, p_loc NUMBER)
IS
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
  INSERT INTO log_table (card_id, location, tran_date)
    VALUES (p_card_id, p_loc, SYSDATE);
  COMMIT;
END log_usage;
```

```
PROCEDURE atm_trans(p_cardnbr NUMBER, p_loc NUMBER) IS
BEGIN
  ... -- try to withdraw cash
  log_usage(p_cardnbr, p_loc);
  ... - if withdrawal fails then
    ROLLBACK;
END atm_trans;
```


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Autonomous Transactions: Example 2

- Your company's website sells music and books.
- A new customer visits the website and decides to buy a music CD.
- The customer types in his or her name, e-mail address, and other personal data, but then decides not to buy the CD after all.
- Maybe you will want to store the customer's personal data anyway, in case he or she comes back to your website in the future.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Autonomous Transactions: Example 2

```
PROCEDURE order_cd (p_name, VARCHAR2, p_email VARCHAR2, p_cd_id
    NUMBER, p_cd_price NUMBER, ... )
IS BEGIN
  INSERT INTO cd_orders (cd_id, cd_price, ...)
    VALUES (p_cd_id, p_cd_price, ...);
  add_new_cust(p_name, p_email);
  IF -- customer decides not to buy THEN ROLLBACK; END IF;
END order_cd;
```

```
PROCEDURE add_new_cust (p_name VARCHAR2, p_email VARCHAR2)
IS
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
  INSERT INTO customers (name, e_mail, ...)
    VALUES (p_name, p_email, ...);
  COMMIT;
END add_new_cust;
```


Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Terminology

Key terms used in this lesson included:

- Autonomous Transaction
- Definer's rights
- Invoker's rights

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Contrast invoker's rights with definer's rights
- Create a procedure that uses invoker's rights
- Create a procedure that executes an Autonomous Transaction

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.


ORACLE
ACADEMY