11

Managing Subprograms

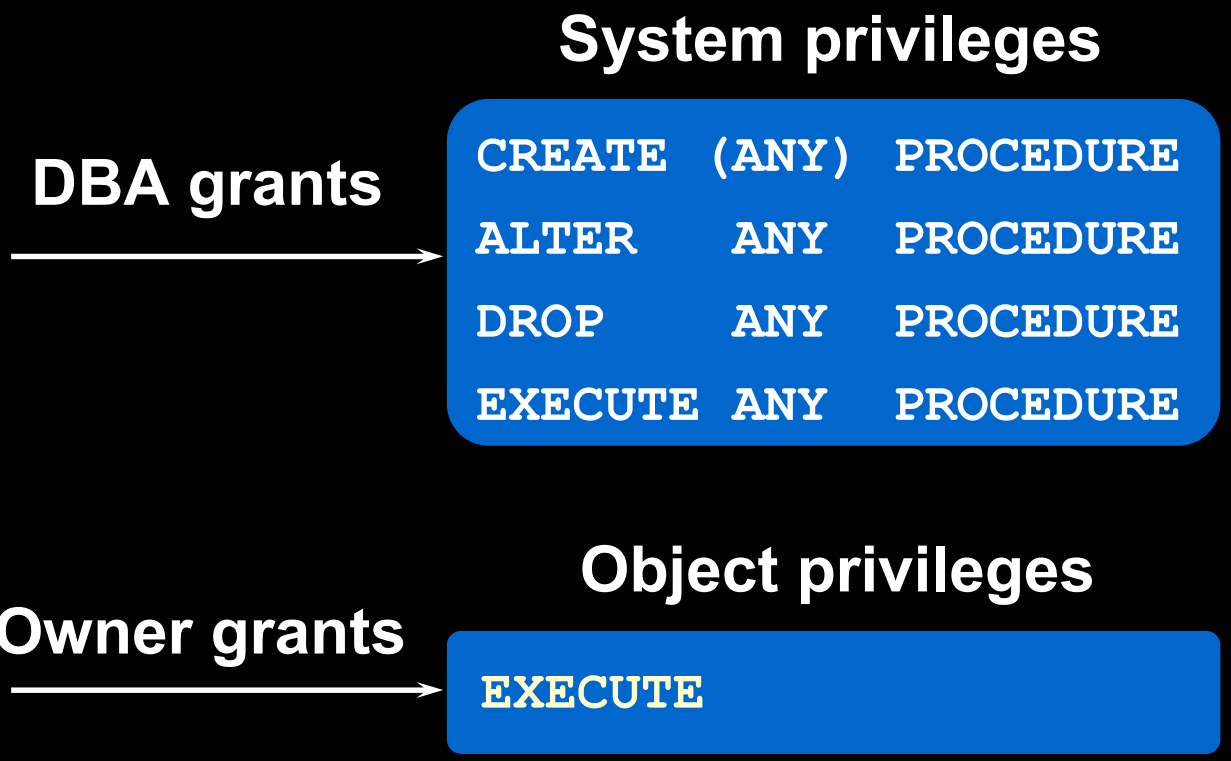
ORACLE

Objectives

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

- Contrast system privileges with object privileges
- Contrast invokers rights with definers rights
- Identify views in the data dictionary to manage stored objects
- Describe how to debug subprograms by using the `DBMS_OUTPUT` package

Required Privileges

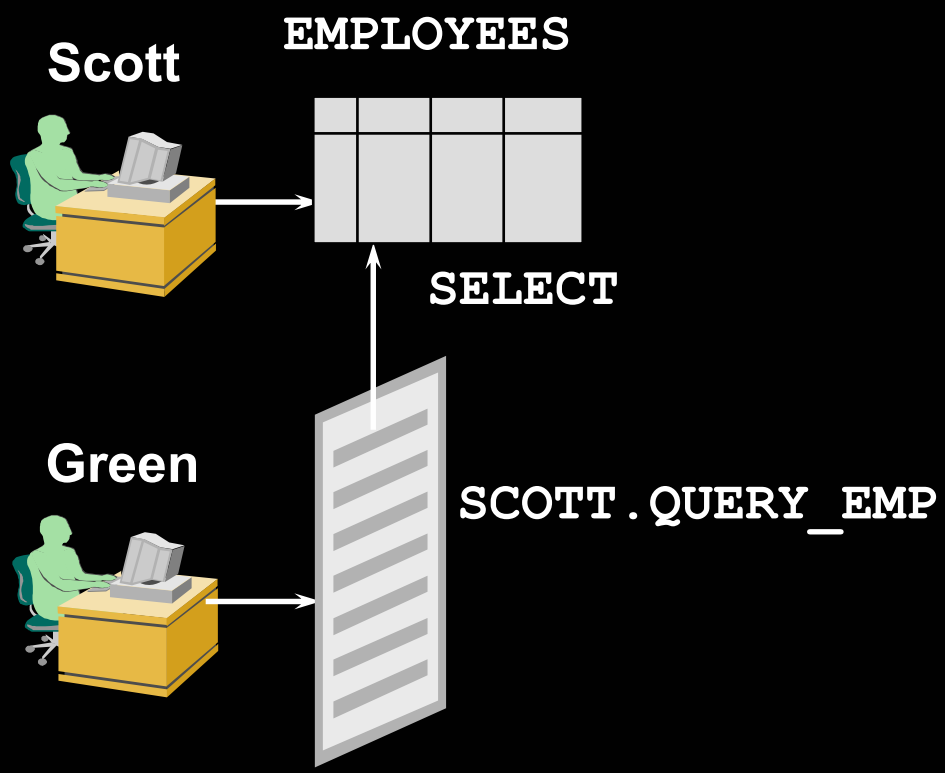


To be able to refer and access objects from a different schema in a subprogram, you must be granted access to the referred objects explicitly, not through a role.

Granting Access to Data

Direct access:

```
GRANT  SELECT
 ON    employees
 TO    scott;
Grant Succeeded.
```



Indirect access:

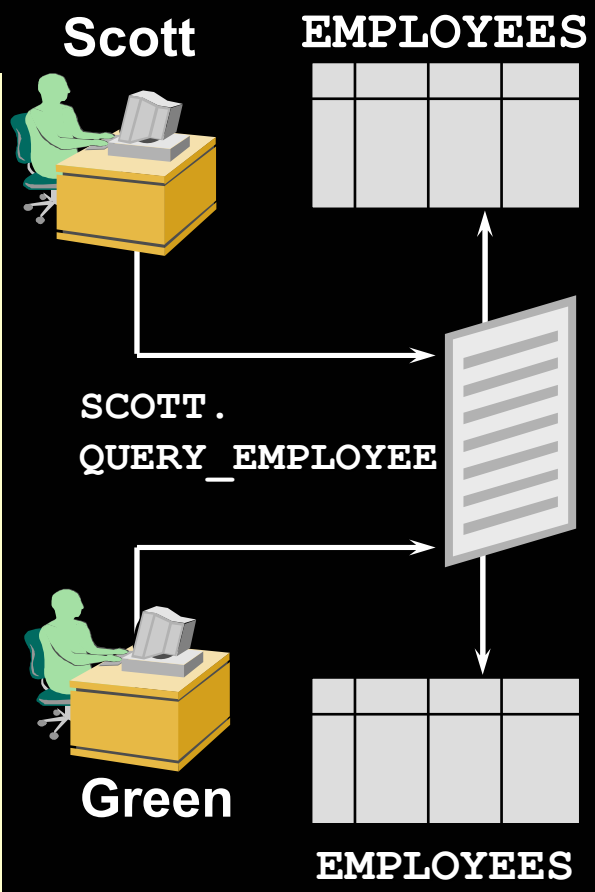
```
GRANT  EXECUTE
 ON    query_emp
 TO    green;
Grant Succeeded.
```

The procedure executes with the privileges of the owner (default).

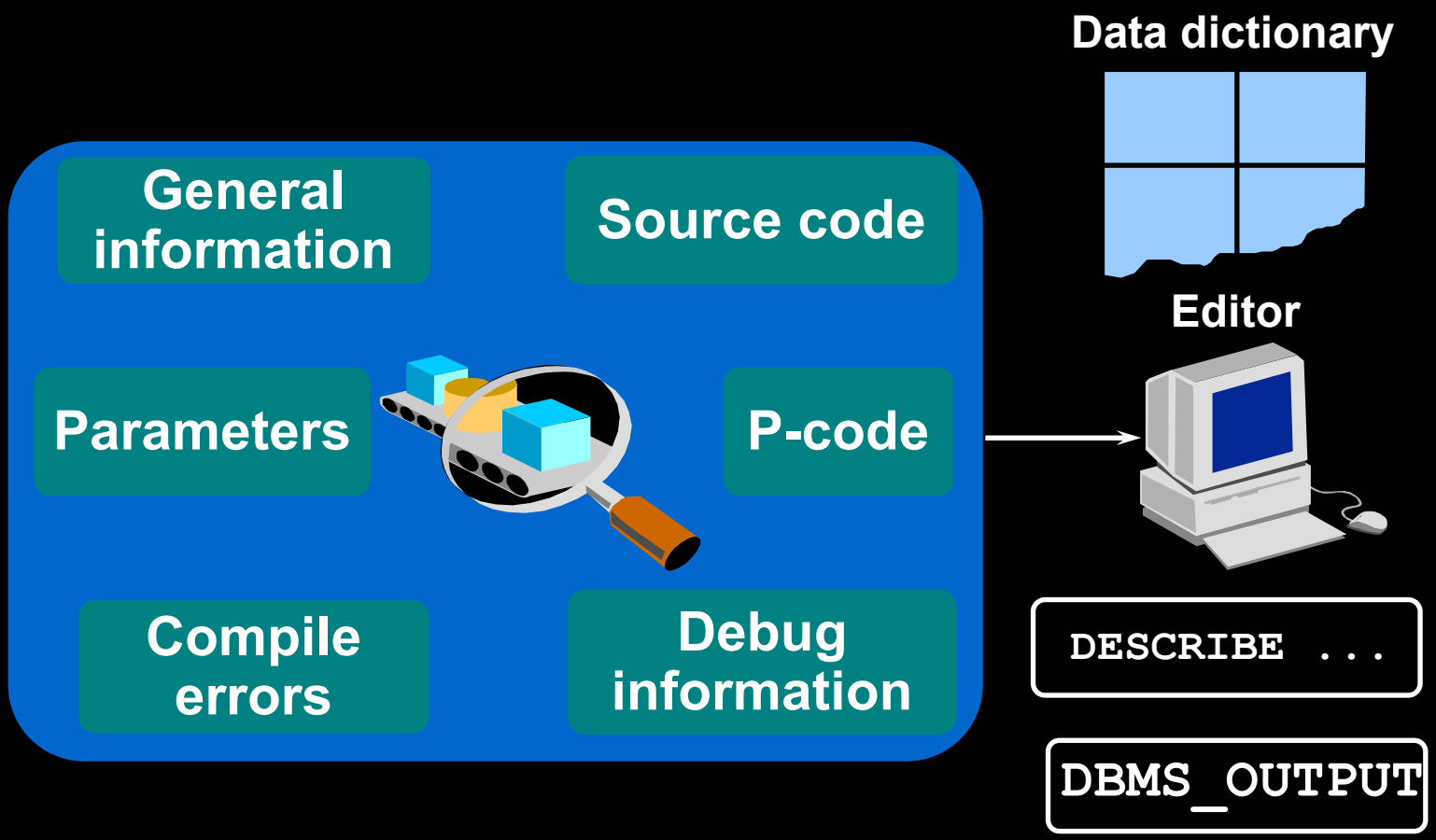
Using Invoker's-Rights

The procedure executes with the privileges of the user.

```
CREATE PROCEDURE query_employee
(p_id IN employees.employee_id%TYPE,
 p_name OUT employees.last_name%TYPE,
 p_salary OUT employees.salary%TYPE,
 p_comm OUT
   employees.commission_pct%TYPE)
AUTHID CURRENT_USER
IS
BEGIN
  SELECT last_name, salary,
         commission_pct
   INTO  p_name, p_salary, p_comm
   FROM  employees
   WHERE employee_id=p_id;
END query_employee;
/
```



Managing Stored PL/SQL Objects



USER_OBJECTS

Column	Column Description
OBJECT_NAME	Name of the object
OBJECT_ID	Internal identifier for the object
OBJECT_TYPE	Type of object, for example, TABLE, PROCEDURE, FUNCTION, PACKAGE, PACKAGE BODY, TRIGGER
CREATED	Date when the object was created
LAST_DDL_TIME	Date when the object was last modified
TIMESTAMP	Date and time when the object was last recompiled
STATUS	VALID or INVALID

*Abridged column list

List All Procedures and Functions

```
SELECT object_name, object_type
FROM  user_objects
WHERE object_type in ('PROCEDURE','FUNCTION')
ORDER BY object_name;
```

OBJECT_NAME	OBJECT_TYPE
ADD_DEPT	PROCEDURE
ADD_JOB	PROCEDURE
ADD_JOB_HISTORY	PROCEDURE
ANNUAL_COMP	FUNCTION
DEL_JOB	PROCEDURE
DML_CALL_SQL	FUNCTION
...	
TAX	FUNCTION
UPD_JOB	PROCEDURE
VALID_DEPTID	FUNCTION

24 rows selected.

USER_SOURCE Data Dictionary View

Column	Column Description
NAME	Name of the object
TYPE	Type of object, for example, PROCEDURE, FUNCTION, PACKAGE, PACKAGE BODY
LINE	Line number of the source code
TEXT	Text of the source code line

List the Code of Procedures and Functions

```
SELECT  text
FROM  user_source
WHERE  name = 'QUERY_EMPLOYEE'
ORDER BY  line;
```

TEXT
PROCEDURE query_employee
(p_id IN employees.employee_id%TYPE, p_name OUT employees.last_name%TYPE,
p_salary OUT employees.salary%TYPE, p_comm OUT employees.commission_pct%TYPE)
AUTHID CURRENT_USER
IS
BEGIN
SELECT last_name, salary, commission_pct
INTO p_name,p_salary,p_comm
FROM employees
WHERE employee_id=p_id;
END query_employee;

11 rows selected.

USER_ERRORS

Column	Column Description
NAME	Name of the object
TYPE	Type of object, for example, PROCEDURE, FUNCTION, PACKAGE, PACKAGE BODY, TRIGGER
SEQUENCE	Sequence number, for ordering
LINE	Line number of the source code at which the error occurs
POSITION	Position in the line at which the error occurs
TEXT	Text of the error message

Detecting Compilation Errors: Example

```
CREATE OR REPLACE PROCEDURE log_execution
IS
BEGIN
INPUT INTO log_table (user_id, log_date)
                                    -- wrong
VALUES (USER, SYSDATE);
END;
/
```

```
Warning: Procedure created with compilation errors.
```

List Compilation Errors by Using USER_ERRORS

```
SELECT line || '/' || position POS, text
FROM   user_errors
WHERE  name = 'LOG_EXECUTION'
ORDER BY line;
```

POS	TEXT
4/7	PLS-00103: Encountered the symbol "INTO" when expecting one of the following: := . (@ % ;
5/1	PLS-00103: Encountered the symbol "VALUES" when expecting one of the following: . (, % ; limit The symbol "VALUES" was ignored.
6/1	PLS-00103: Encountered the symbol "END"

List Compilation Errors by Using SHOW ERRORS

```
SHOW ERRORS PROCEDURE log_execution
```

Errors for PROCEDURE LOG_EXECUTION:

LINE/COL	ERROR
4/7	PLS-00103: Encountered the symbol "INTO" when expecting one of th e following: := . (@ % ;
5/1	PLS-00103: Encountered the symbol "VALUES" when expecting one of the following: . (, % ; limit The symbol "VALUES" was ignore d.
6/1	PLS-00103: Encountered the symbol "END"

DESCRIBE in *i*SQL*Plus

```
DESCRIBE query_employee
DESCRIBE add_dept
DESCRIBE tax
```

PROCEDURE **QUERY_EMPLOYEE**

Argument Name	Type	In/Out	Default?
P_ID	NUMBER(6)	IN	
P_NAME	VARCHAR2(25)	OUT	
P_SALARY	NUMBER(8,2)	OUT	
P_COMM	NUMBER(2,2)	OUT	

PROCEDURE ADD_DEPT

Argument Name	Type	In/Out	Default?
P_NAME	VARCHAR2(30)	IN	DEFAULT
P_LOC	NUMBER(4)	IN	DEFAULT

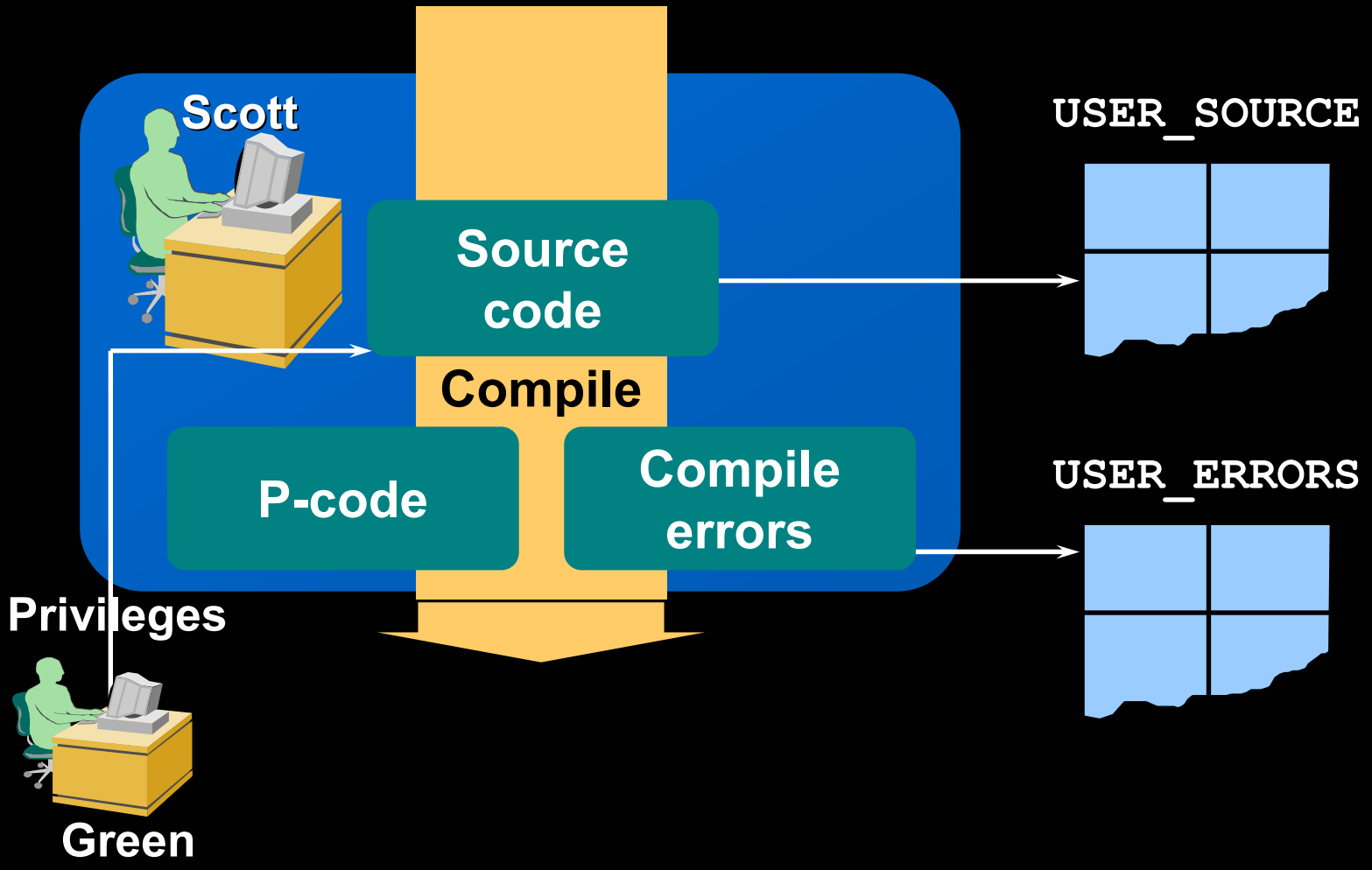
FUNCTION TAX RETURNS NUMBER

Argument Name	Type	In/Out	Default?
P_VALUE	NUMBER	IN	

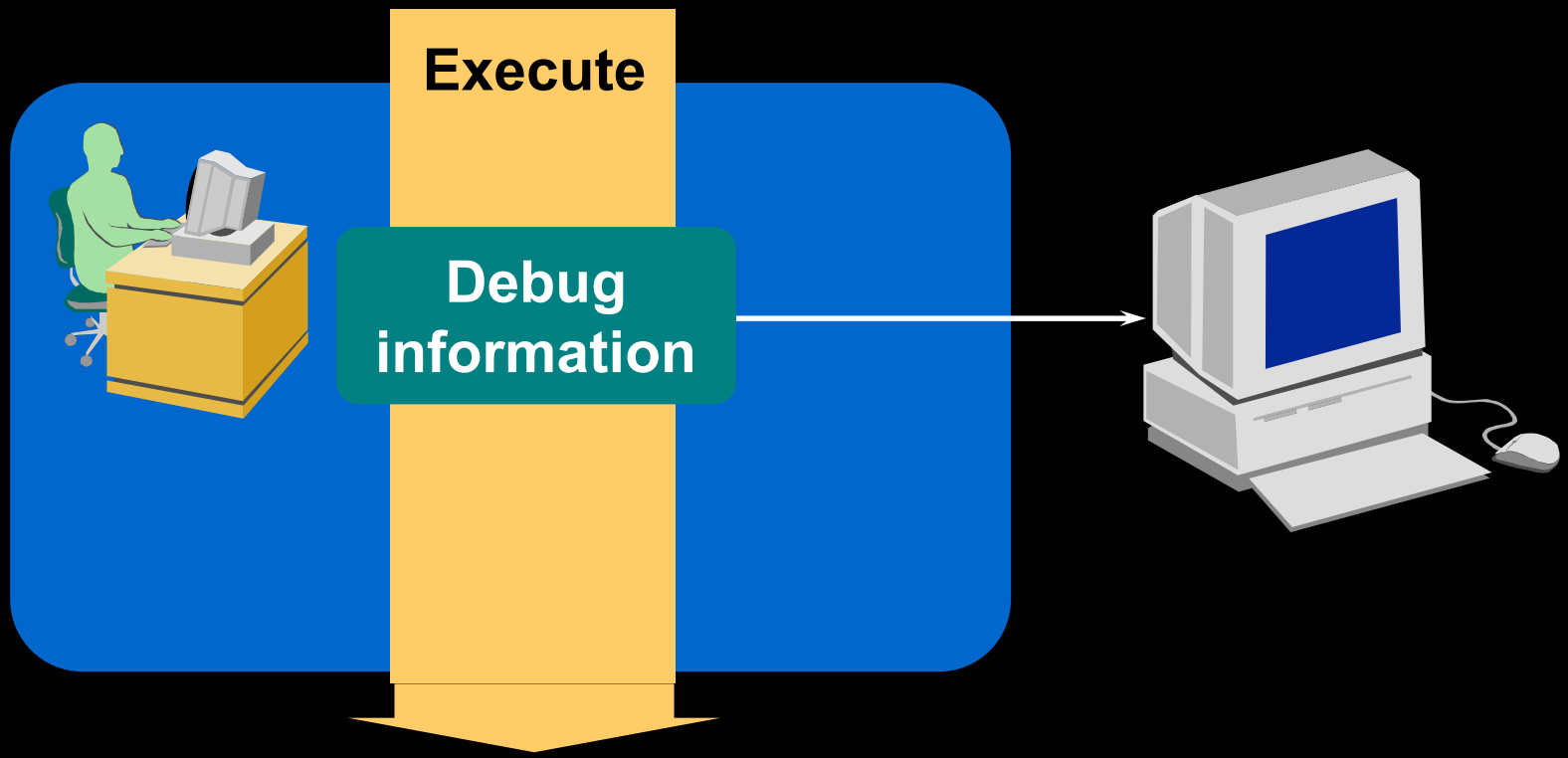
Debugging PL/SQL Program Units

- The `DBMS_OUTPUT` package:
  - Accumulates information into a buffer
  - Allows retrieval of the information from the buffer
- Autonomous procedure calls (for example, writing the output to a log table)
- Software that uses `DBMS_DEBUG`
  - Procedure Builder
  - Third-party debugging software

Summary



Summary



Practice 11 Overview

This practice covers the following topics:

- Re-creating the source file for a procedure
- Re-creating the source file for a function

Latihan bab 26 – Managing Subprograms

1. Buka terminal SQL Plus, kemudian login ke database oracle menggunakan user **HR**, kemudian jalankan kode program berikut:

```
CREATE OR REPLACE PROCEDURE query_emp
   (e_id employees.employee_id%TYPE default 100)
IS
   e_last_name employees.last_name%TYPE;
   e_salary employees.salary%TYPE;
BEGIN
   SELECT last_name, salary INTO e_last_name, e_salary
   FROM employees
   WHERE employee_id = e_id;
   dbms_output.put_line(e_last_name,  e_salary);
END query_emp;
/
```

2. Query dictionary view **USER_OBJECTS**, catat status dari procedure yang dibuat pada nomor 1!
3. Query dictionary view **USER_ERRORS**, catat nomor baris dimana terdapat error!
4. Query dictionary view **USER_SOURCE**, lihat baris yang mengandung error, kemudian perbaiki procedure yang telah dibuat pada nomor 1!
5. Buka terminal SQL Plus kedua dan login dengan user **SCOTT**. Kemudian jalankan perintah:

```
SQL> SET SERVEROUTPUT ON
SQL> EXEC hr.query_emp;
```

6. Pada terminal SQL Plus pertama, beri hak **EXECUTE** pada procedure yang dibuat di nomor 1 kepada user **SCOTT**.
7. Pada terminal kedua, jalankan perintah:

```
SQL> EXEC hr.query_emp;
```

   Apa yang terjadi ?
8. Ubah procedure pada nomor satu dengan menambahkan baris berikut sebelum statemen '**IS**'

```
AUTHID CURRENT_USER
```

   Jalankan lagi kode program tersebut pada terminal SQL Plus yg pertama!
9. Dari terminal kedua, jalankan perintah berikut:

```
SQL> EXEC hr.query_emp;
```

   Apa yang terjadi ? Kenapa ?