SUBPROGRAMS

Overview of Subprograms

A subprogram:
Is a named PL/SQL block that can accept parameters
and be invoked from a calling environment

Is of two types:
- A procedure that performs an action
- A function that computes a value

Is based on standard PL/SQL block structure

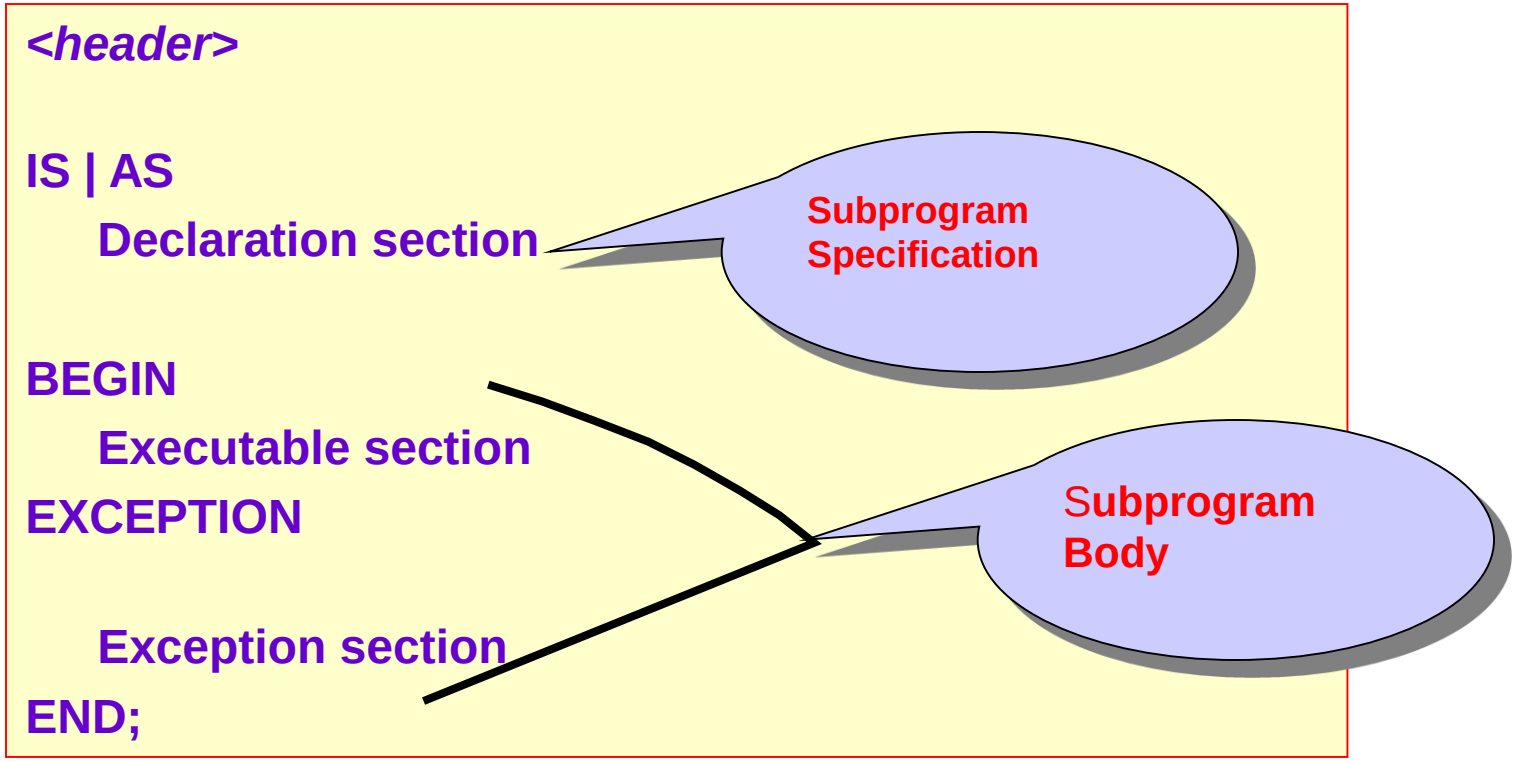
Provides modularity, reusability, extensibility,
and maintainability

Provides easy maintenance, improved data security and integrity,
improved performance, and improved code clarity

Benefits of Subprograms

- Easy maintenance
- Improved data security and integrity
- Improved performance
- Improved code clarity

Block Structure for PL/SQL Subprograms



Stored Procedures

Definition of a Procedure

A procedure is a named PL/SQL block that can accept parameters (sometimes referred to as arguments), and be invoked.

In general, create a procedure to perform an action. A procedure has a header, a declaration section, an executable section, and an optional exception-handling section.

A procedure can be compiled and stored in the database as a schema object.

Procedures promote reusability and maintainability. When validated, they can be used in any number of applications. If the requirements change, only the procedure needs to be updated.

Stored Procedures

Syntax of creating a procedure

```
CREATE [OR REPLACE] PROCEDURE procedure_name
 (parameter_name [mode] datatype,
  parameter_name [mode] datatype,
  . . .)
IS|AS
      --- Declare Section
Begin

      --- Procedure Body

End [procedure_name] ;
```

The REPLACE option indicates that if the procedure exists, it will be dropped and replaced with the new version created by the statement.

Note: Use SHOW ERRORS to view compilation errors.

Parameters

Formal Versus Actual Parameters

Formal parameters: variables declared in the parameter list of a subprogram specification

Example:

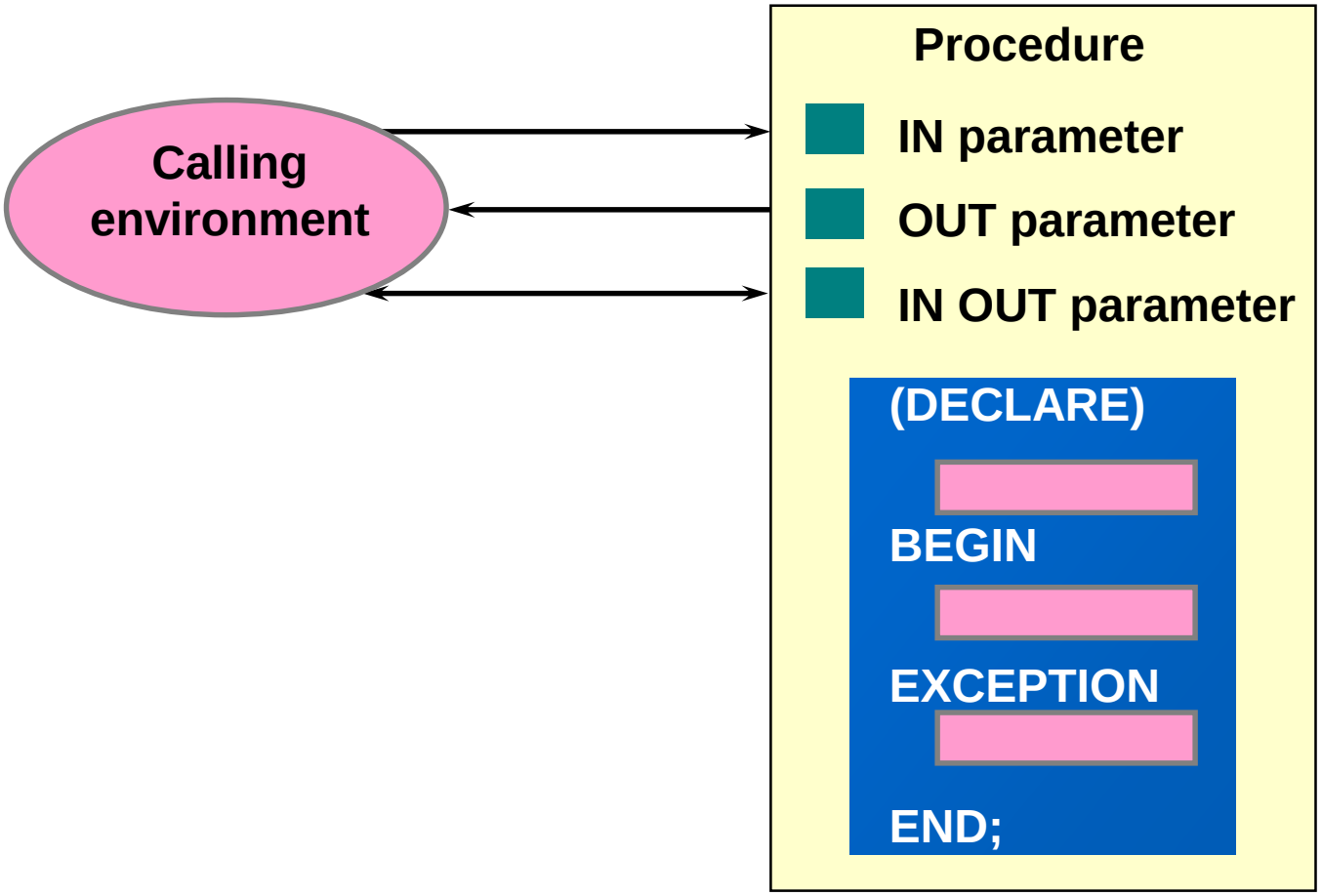
```
CREATE PROCEDURE inc_salary(p_id NUMBER, p_amount
NUMBER)

...

END inc_salary;
```

Actual parameters: variables or expressions referenced in the parameter list of a subprogram call

Example:

```
inc_salary(v_id, 2000)
```

Parameter Modes



Parameter Modes

IN	OUT	IN OUT
Default mode	Must be specified	Must be specified
Value is passed into subprogram	Returned to calling environment	Passed into subprogram; returned to calling environment
Formal parameter acts as a constant	Uninitialized variable	Initialized variable
Actual parameter can be a literal, expression, constant, or initialized variable	Must be a variable	Must be a variable
Can be assigned a default value p_name varchar2 default 'TEMP'	Cannot be assigned a default value	Cannot be assigned a default value

Stored Function

Syntax of creating a function

```
CREATE [OR REPLACE] FUNCTION function_name
 (parameter_name [mode] datatype,
  parameter_name [mode] datatype,
  . . .)  RETURN return_type
IS|AS
        --- Declare Section
Begin
 -----
 return <value>;

End [function_name] ;
```

Note: Avoid using OUT and IN OUT mode in functions

Thank You!