Creating Functions

Overview of Stored Functions

A function:

- Is a named PL/SQL block that returns a value
- Can be stored in the database as a schema object for repeated execution
- Is called as part of an expression or is used to provide a parameter value

In general to Compute A Value

Examples :

- we can crate function to return the salary for an employee
- Function to retrieve the full name for the employee
- Function to calculate the GPA for the Student
- Function to compute the tax for a salary

Creating Functions

The PL/SQL block must have at least one `RETURN` statement.

```
CREATE [OR REPLACE] FUNCTION function_name
 [(parameter1 [mode1] datatype1, . . .)]
RETURN datatype IS|AS
 [local_variable_declarations;
  . . .]
BEGIN
  -- actions;
  RETURN expression;
END [function_name];
```

PL/SQL Block

should be same Data type

- Host variables not Allowed, also substitute variables &
- It should be at least one return expression in executable section
- Return datatype should be without size.
- Out / IN OUT can be used , but this not good Practice

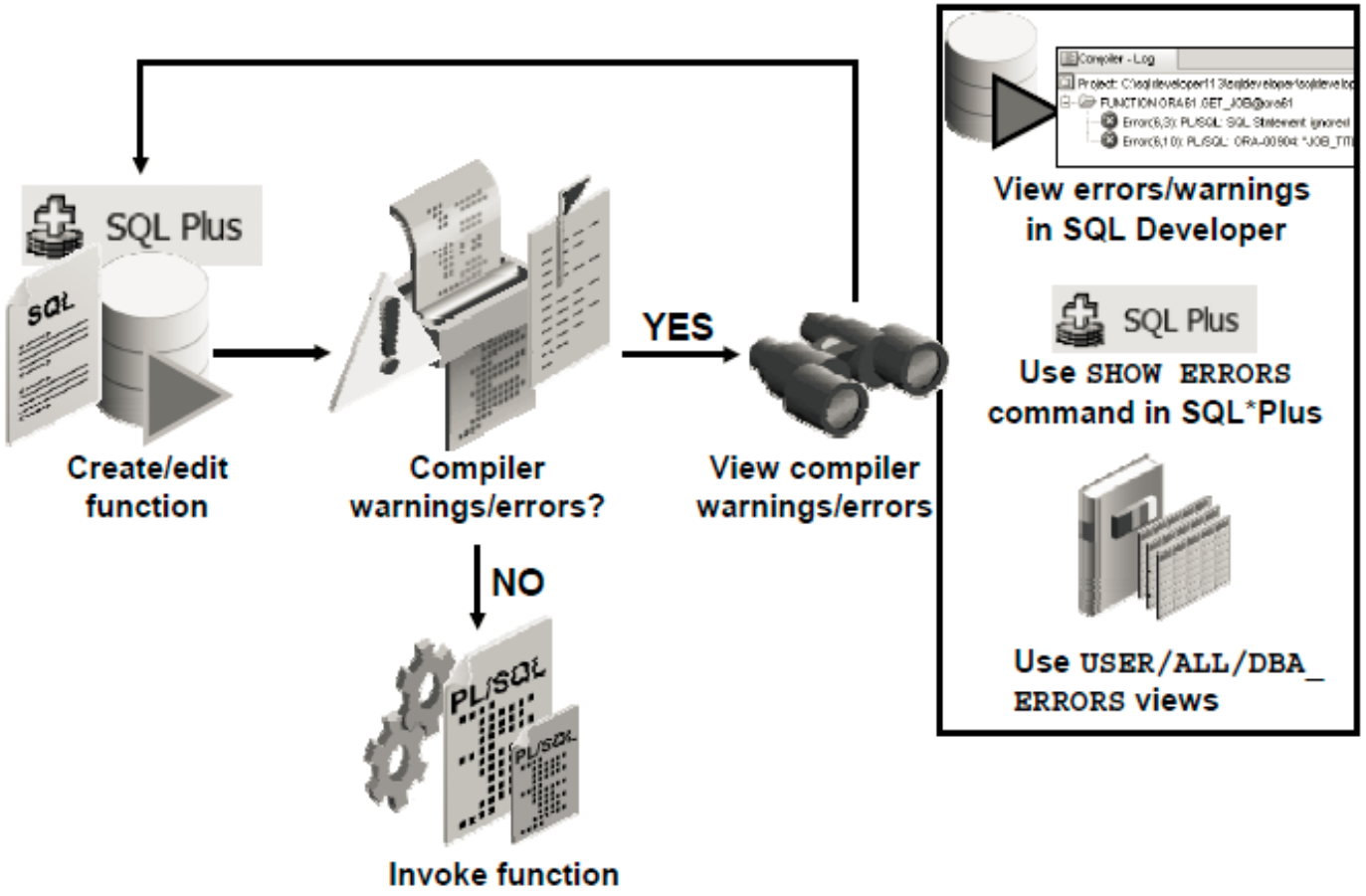
The Difference Between Procedures and Functions

Procedures	Functions
Execute as a PL/SQL statement	Invoke as part of an expression
Do not contain RETURN clause in the header	Must contain a RETURN clause in the header
Can pass values (if any) using output parameters	Must return a single value
Can contain a RETURN statement without a value	Must contain at least one RETURN statement

To Preform an Action	To return A value
Can not be used in select	Can be used in Select But it should not include OUT/ IN OUT Parameters

A Procedure that have one Parameter(OUT) would be better rewritten as A function

Creating and Running Functions: Overview



maxvlearn

Function Example

```
create or replace function get_sal
(p_emp_id number)
return number
is
v_sal number;
begin
   select salary into v_sal
   from employees
   where employee_id=p_emp_id;

   return v_sal;

end;
```

Advantages of User-Defined Functions in SQL Statements

- Can extend SQL where activities are too complex, too awkward, or unavailable with SQL
- Can increase efficiency when used in the `WHERE` clause to filter data, as opposed to filtering the data in the application
- Can manipulate data values

Calling User-Defined Functions in SQL Statements

User-defined functions act like built-in single-row functions and can be used in:

- The `SELECT` list or clause of a query
- Conditional expressions of the `WHERE` and `HAVING` clauses
- The `CONNECT BY`, `START WITH`, `ORDER BY`, and `GROUP BY` clauses of a query
- The `VALUES` clause of the `INSERT` statement
- The `SET` clause of the `UPDATE` statement

Restrictions When Calling Functions from SQL Expressions

- User-defined functions that are callable from SQL expressions must:
  - Be stored in the database
  - Accept only `IN` parameters with valid SQL data types, not PL/SQL-specific types (Record, table , Boolean)
  - Return valid SQL data types, not PL/SQL-specific types
- When calling functions in SQL statements:
  - Parameters must be specified with positional notation
  - You must own the function or have the `EXECUTE` privilege

This Before 11g only

- Can not be used in Check constraint (create table/ alter table)
- Can not be used as default value for a column

maxvlearn

Controlling Side Effects When Calling Functions from SQL Expressions

Functions called from:

- A `SELECT` statement cannot contain DML statements
- An `UPDATE` or `DELETE` statement on a table `T` cannot query or contain DML on the same table `T`
- SQL statements cannot end transactions (that is, cannot execute `COMMIT` or `ROLLBACK` operations)

Note: Calls to subprograms that break these restrictions are also not allowed in the function.

When a function is called from update/ delete , then then the function can not Query or modify database tables modified by that statement
Error: mutating table

Thank You