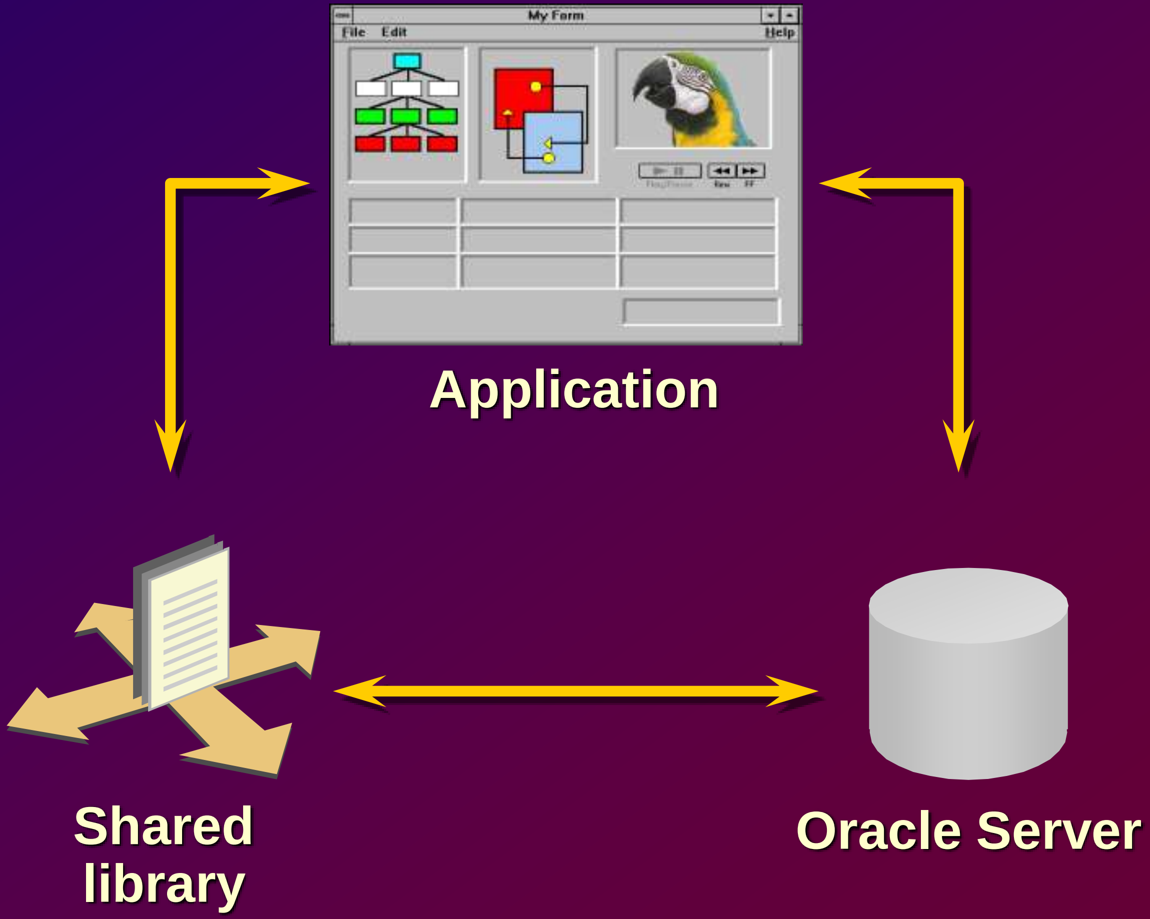
About PL/SQL

- PL/SQL is an extension to SQL with design features of programming languages.
- Data manipulation and query statements of SQL are included within procedural units of code.

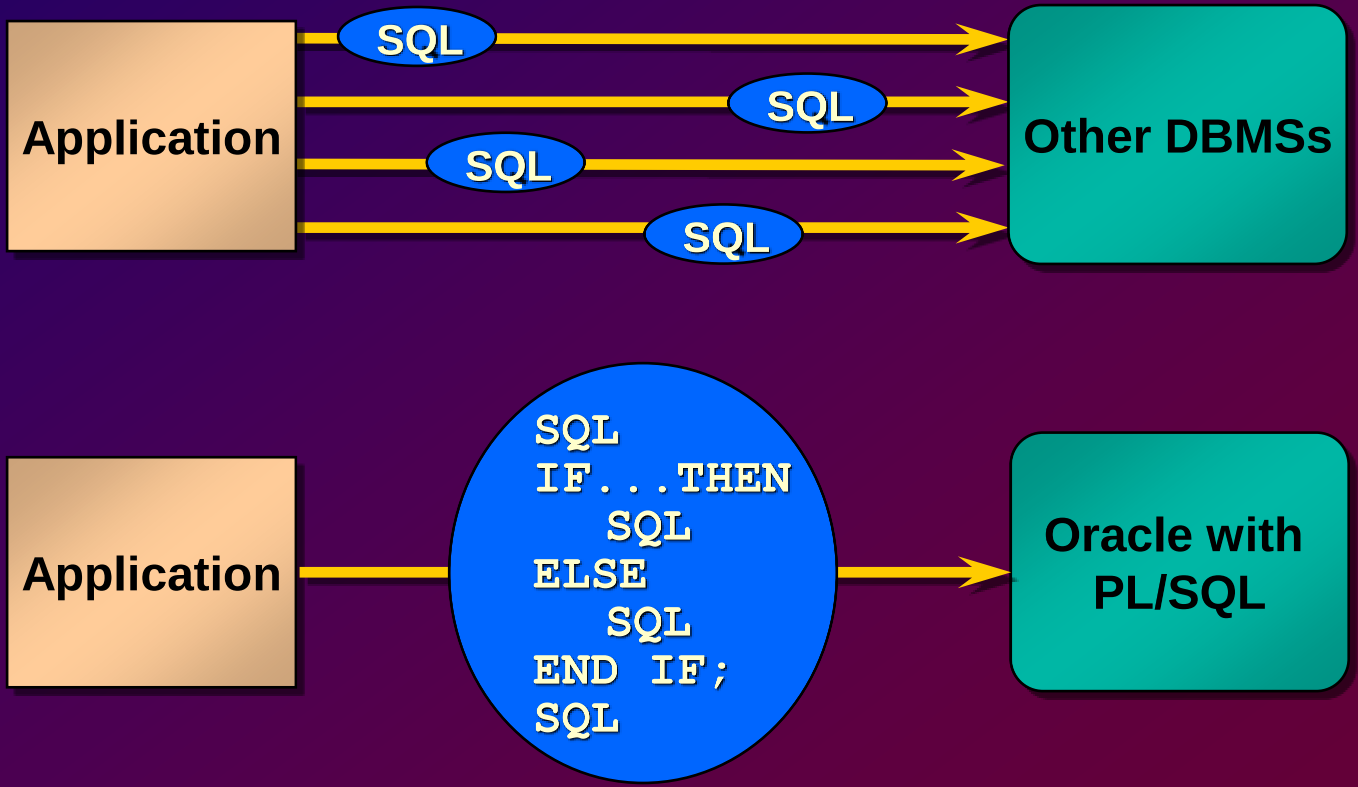
Benefits of PL/SQL

Integration



Benefits of PL/SQL

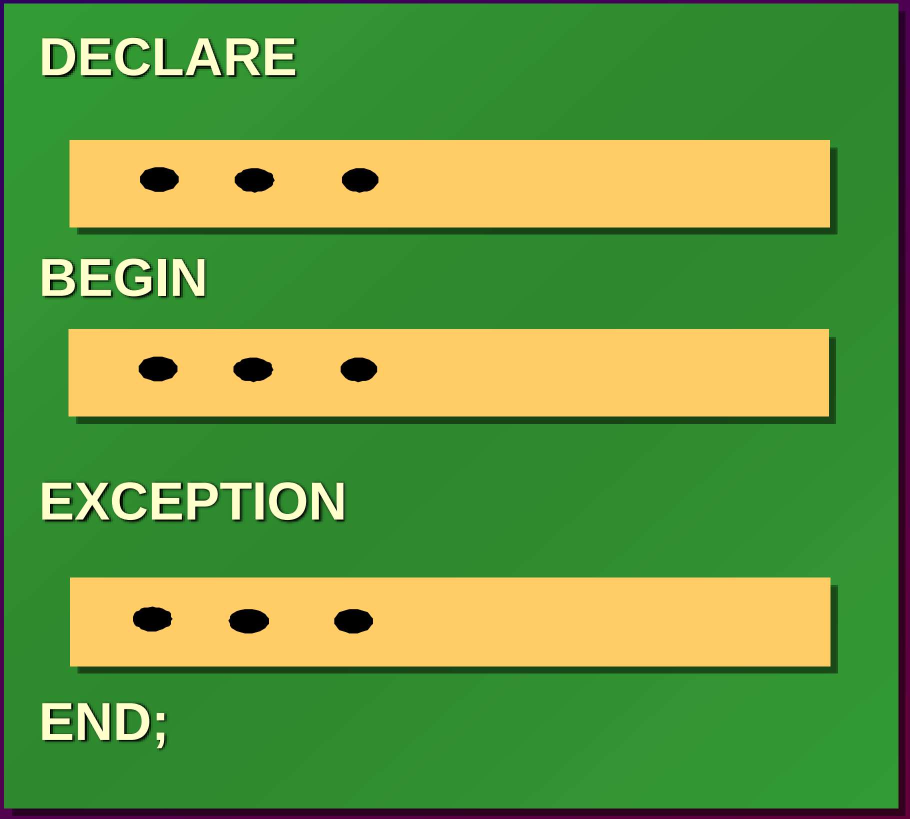
Improved Performance



```
SQL
IF...THEN
    SQL
ELSE
    SQL
END IF;
SQL
```

Benefits of PL/SQL

Modularize program development



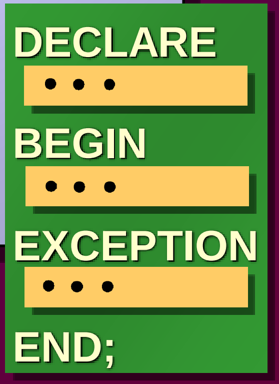
```
DECLARE
    • • •
BEGIN
    • • •
EXCEPTION
    • • •
END;
```

Benefits of PL/SQL

- You can program with procedural language control structures.
- It can handle errors.
- It is portable.
- You can declare identifiers.

PL/SQL Block Structure

- DECLARE – Optional
  - Variables, cursors, user-defined exceptions
- BEGIN – Mandatory
  - SQL statements
  - PL/SQL statements
- EXCEPTION – Optional
  - Actions to perform when errors occur
- END; – Mandatory



Block Types

Anonymous

```
 [DECLARE]


BEGIN
  --statements

 [EXCEPTION]


END;
```

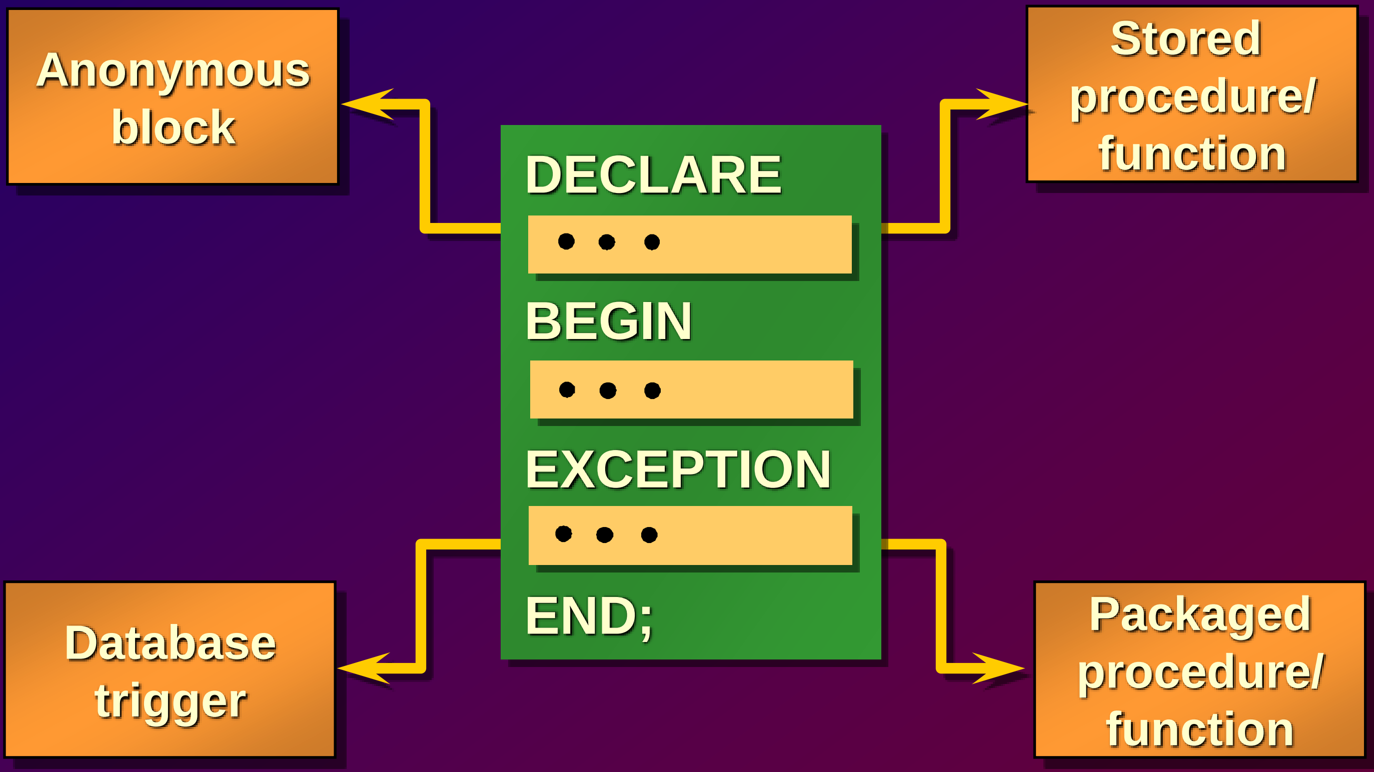
Procedure

```
PROCEDURE name
IS

BEGIN
  --statements

 [EXCEPTION]


END;
```

Function

```
FUNCTION name
RETURN datatype
IS
BEGIN
  --statements
  RETURN value;
 [EXCEPTION]


END;
```

Program Constructs



Use of Variables

Use variables for:

- Temporary storage of data
- Manipulation of stored values
- Reusability
- Ease of maintenance

Handling Variables in PL/SQL

- Declare and initialize variables in the declaration section.
- Assign new values to variables in the executable section.
- Pass values into PL/SQL blocks through parameters.
- View results through output variables.

Types of Variables

- PL/SQL variables:
  - Scalar
  - Composite
  - Reference
  - LOB (large objects)
- Non-PL/SQL variables: Bind and host variables

Declaring PL/SQL Variables

Syntax

```
identifier [CONSTANT] datatype [NOT NULL]
     [:= | DEFAULT expr];
```

Examples

```
Declare
  v_hiredate       DATE;
  v_deptno         NUMBER(2) NOT NULL := 10;
  v_location       VARCHAR2(13) := 'Atlanta';
  c_comm           CONSTANT NUMBER := 1400;
```

Declaring PL/SQL Variables

Guidelines

- **Follow naming conventions.**
- **Initialize variables designated as NOT NULL.**
- **Initialize identifiers by using the assignment operator (:=) or the DEFAULT reserved word.**

Naming Rules

- Two variables can have the same name, provided they are in different blocks.
- The variable name (identifier) should not be the same as the name of table columns used in the block.

```
DECLARE
  empno  NUMBER(4);
BEGIN
  SELECT     empno
  INTO       empno
  FROM       emp
  WHERE      ename = 'SMITH';
END;
```

Adopt a naming convention for PL/SQL identifiers: for example, v_empno

Assigning Values to Variables

Syntax

```
identifier := expr;
```

Examples

Set a predefined hiredate for new employees.

```
v_hiredate := '31-DEC-98';
```

Set the employee name to “Maduro.”

```
v_ename := 'Maduro';
```

Variable Initialization and Keywords

Using:

- Assignment operator (:=)
- DEFAULT keyword
- NOT NULL constraint

Base Scalar Datatypes

- VARCHAR2 (*maximum_length*)
- NUMBER [(*precision, scale*)]
- DATE
- CHAR [(*maximum_length*)]
- LONG
- LONG RAW
- BOOLEAN
- BINARY_INTEGER
- PLS_INTEGER

Scalar Variable Declarations

Examples

```
v_job           VARCHAR2(9);
v_count         BINARY_INTEGER := 0;
v_total_sal     NUMBER(9,2) := 0;
v_orderdate     DATE := SYSDATE + 7;
c_tax_rate      CONSTANT NUMBER(3,2) := 8.25;
v_valid         BOOLEAN NOT NULL := TRUE;
```

The %TYPE Attribute

- Declare a variable according to:
  - A database column definition
  - Another previously declared variable
- Prefix %TYPE with:
  - The database table and column
  - The previously declared variable name

Declaring Variables with the %TYPE Attribute

Examples

```
...
  v_ename                  emp.ename%TYPE;
  v_balance                NUMBER(7,2);
  v_min_balance            v_balance%TYPE := 10;
...
```

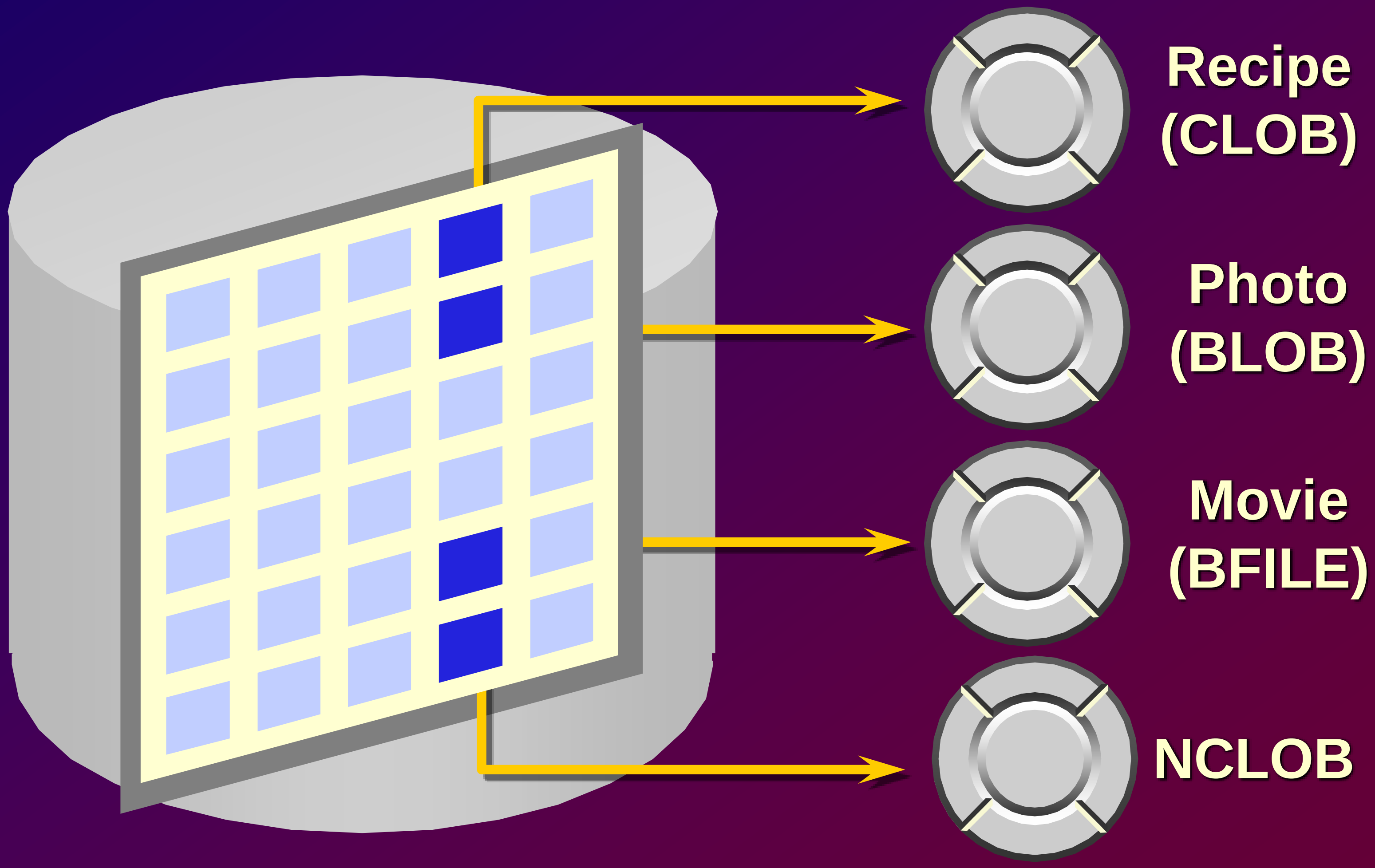
Declaring Boolean Variables

- Only the values TRUE, FALSE, and NULL can be assigned to a Boolean variable.
- The variables are connected by the logical operators AND, OR, and NOT.
- The variables always yield TRUE, FALSE, or NULL.
- Arithmetic, character, and date expressions can be used to return a Boolean value.

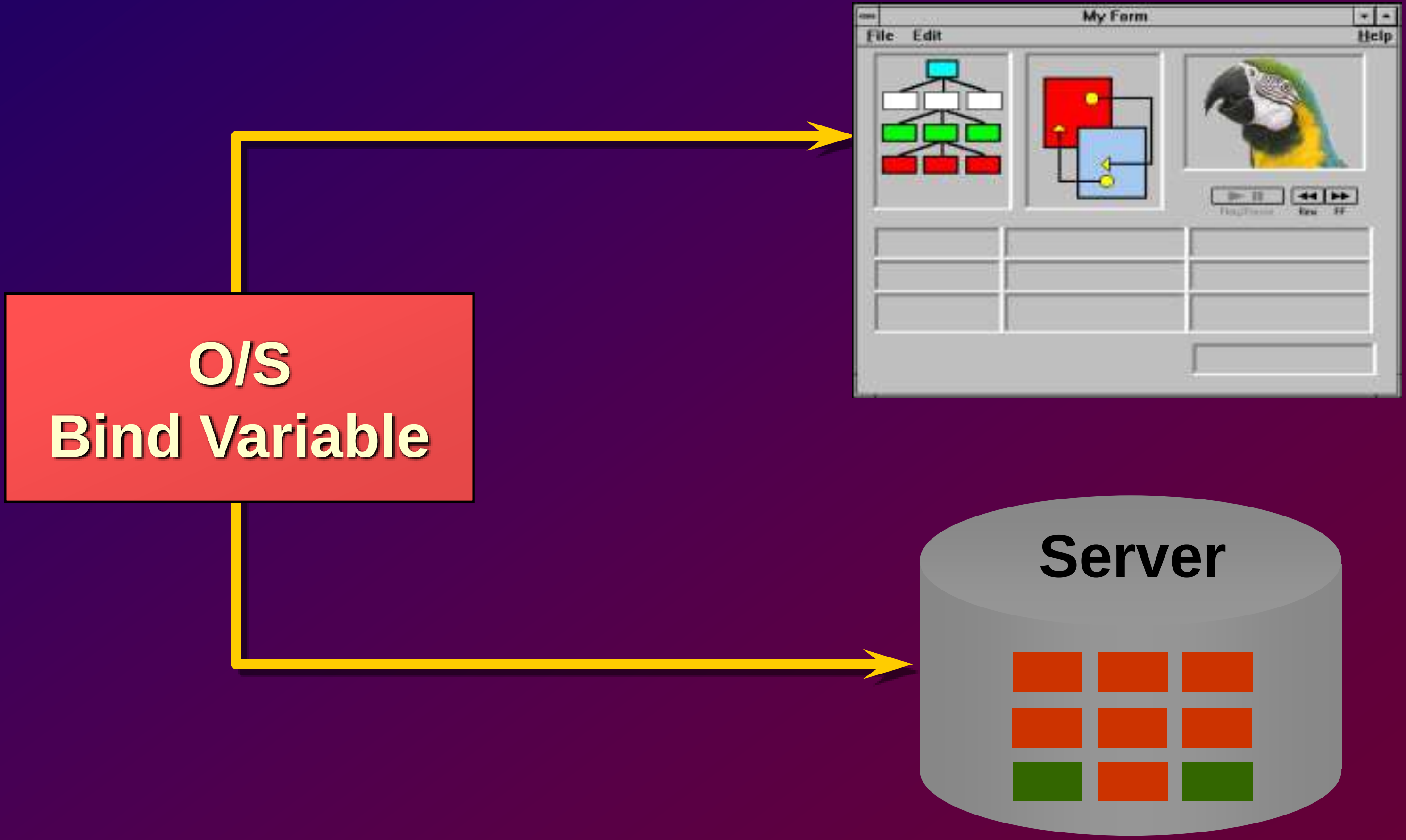
Composite Datatypes

- PL/SQL TABLES
- PL/SQL RECORDS

LOB Datatype Variables



Bind Variables



Referencing Non-PL/SQL Variables

Store the annual salary into a SQL*Plus host variable.

Reference non-PL/SQL variables as host variables.

- Prefix the references with a colon (:).

```
Variable v number;
BEGIN
 :v := v_sal / 12;
END;
/
Print v;
```

DBMS_OUTPUT.PUT_LINE

- An Oracle-supplied packaged procedure
- An alternative for displaying data from a PL/SQL block
- Must be enabled in SQL*Plus or SqlDeveloper with

```
SET SERVEROUTPUT ON
```