University
Politehnica
of Bucharest

PL/SQL (I)

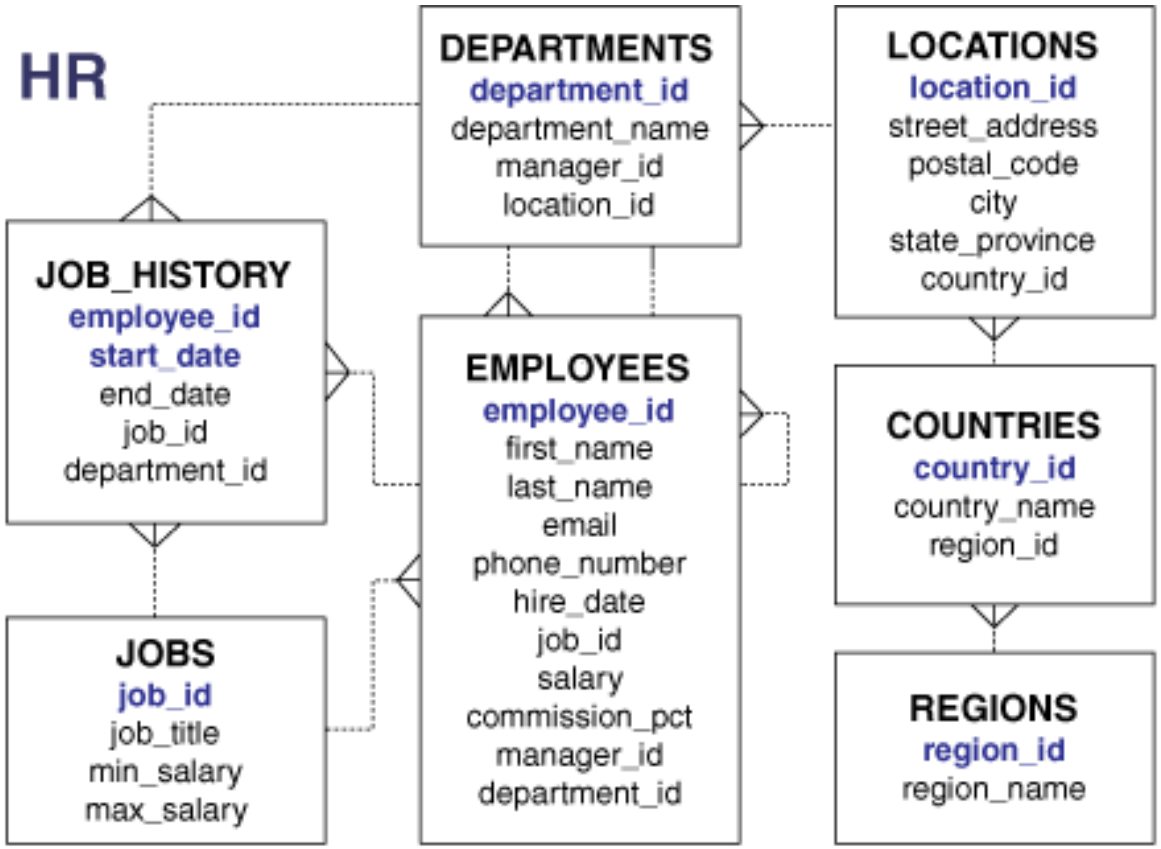
Ș.L. Dr. Ing. Ciprian-Octavian Truică
ciprian.truica@upb.ro

Overview

- PL/SQL
- PL/SQL Main Features
- PL/SQL Engine

Examples

- Database



Overview

- PL/SQL
- PL/SQL Main Features
- PL/SQL Engine

PL/SQL

- PL/SQL (Procedural Language/SQL)
  - Is the Oracle procedural extension of SQL
  - Is a portable, high-performance transaction-processing language
  - Is a proprietary language

PL/SQL

- Advantages of PL/SQL:
  - Tight Integration with SQL
  - High Performance
  - High Productivity
  - Portability
  - Scalability
  - Manageability
  - Support for Object-Oriented Programming

PL/SQL
Tight Integration with SQL

- PL/SQL supports
  - SQL data manipulation statements DML
  - Cursor control statements (OPEN, LOOP, CLOSE)
  - Transaction control statements TCL

PL/SQL
Tight Integration with SQL

- DML (Data Manipulation Language)
  - SELECT
  - INSERT
  - UPDATE
  - DELETE
- TCL (Transaction Control Language)
  - COMMIT
  - ROLLBACK
  - SAVEPOINT

PL/SQL
Tight Integration with SQL

- PL/SQL supports
  - All Oracle SQL Functions
  - All SQL Operators
  - Pseudocolumns
    - A **pseudocolumn** behaves like a table column but is not actually stored in the table
    - E.g., COLUMN_VALUE, **ROWID**, **ROWNUM**, XMLDATA

PL/SQL
Tight Integration with SQL

- PL/SQL supports all SQL data types
  - No need for conversions between PL/SQL and SQL data types
  - Attributes for data type correlation
    - %ROWTYPE attribute to declare a record that represents either a full or partial row of a database table or view
    - %TYPE attribute to declare a data item of the same data type as a previously declared variable or column

PL/SQL
Tight Integration with SQL

- PL/SQL supports
  - Running SQL query
  - Processing the rows of the result set one at a time

- PL/SQL functions can be declared and defined in the WITH clauses of SQL SELECT statements (CTE – Common Table Expressions)

Examples

```sql
with no_emps as (
    select department_id, count(*) c
    from employees
    group by department_id
)
select d.department_name, q.c  emps
from departments d
    inner join no_emps q
        on q.department_id = d.department_id
order by 1;

-- Equivalent

select
    d.department_name,
    count(e.employee_id) emps
from departments d
    inner join employees e
        on d.department_id=e.department_id
group by d.department_name
order by 1;
```

PL/SQL
Tight Integration with SQL

- PL/SQL supports
  - Static SQL
    - SQL whose full text is known at compile time
  - Dynamic SQL
    - SQL whose full text is not known until run time
    - Make applications more
      - Flexible
      - Versatile

PL/SQL
Tight Integration with SQL

- Dynamic SQL Drawbacks
  - Performance loss
    - The execution plan for dynamic queries cannot be cached
  - Hard to debug
    - The error management becomes more unreliable.
    - Maintenance is difficult because the schema is hard coded in the dynamic code.
  - Security can be compromised with SQL injection

PL/SQL
High Performance

- PL/SQL
  - Statements can be organized into blocks (BEGIN … END)
  - Blocks significantly reduce traffic between the application and the database.
- High performance is achieved through:
  - Bind Variables
  - Subprograms
  - Optimizer

PL/SQL
High Performance

- Bind Variables
  - Are placeholders for actual values in SQL statements
  - Allow the database server to prepare the statement once and execute it multiple times without reparsing or reanalyzing it.
  - The PL/SQL compiler turns the variables in the WHERE and VALUES clauses into bind variables when embedding a SQL data manipulation statements directly in a block
  - Oracle Database can reuse these SQL statements each time the same code runs
  - PL/SQL does not create bind variables automatically when you use dynamic SQL
    - They can be used with dynamic SQL by specifying them explicitly

PL/SQL
High Performance

- PL/SQL supports subprograms:
  - Functions
  - Procedures
- Subprograms
  - Are stored in executable form
  - Can be invoked repeatedly
  - Are cached lowering memory requirements
  - Are shared among users lowering invocation overhead
  - A single invocation over the network can start a large job
    - Reducing network traffic
    - Improving response times
- Optimizer
  - The PL/SQL compiler has an optimizer that can rearrange code for better performance.

PL/SQL
High Productivity

- PL/SQL
  - Is the same in all environments
  - Can be used to write compact code for manipulating data
  - Is a scripting language
  - Can query, transform, and update data in a database
  - Supported by many other Oracle tools (Forms Builder, Apex, etc.)

PL/SQL Portability

- PL/SQL is a portable and standard language for Oracle development
- PL/SQL applications can be run on any operating system and platform where Oracle Database runs
- PL/SQL applications can be run on different environments (e.g., server, client application, etc.)

PL/SQL Scalability

- PL/SQL stored subprograms increase scalability by centralizing application processing on the database server
- The shared memory facilities of the shared server let Oracle Database support thousands of concurrent users on a single node
- Oracle Connection Manager can be used to multiplex network connections

PL/SQL Manageability

- PL/SQL stored subprograms increase manageability:
  - Only one copy of a subprogram can be maintained on the database server
  - Rather than one copy on each client system
- Any number of applications can use the subprograms
- Subprograms can be changed without affecting the applications that invoke them

PL/SQL
Support for Object-Oriented Programming

- PL/SQL supports OOP
  - Allows defining object types that can be used in object-oriented designs
  - Abstract Data Types (ADT):
    - Consists of a data structure and subprograms that manipulate the data
    - The variables that form the data structure are called attributes.
    - The subprograms that manipulate the attributes are called methods.

Overview

- PL/SQL
- PL/SQL Main Features
- PL/SQL Engine

PL/SQL Main Features

- PL/SQL combines the data-manipulating power of SQL with the processing power of procedural languages
- SQL statements can be issued from a PL/SQL program
- Complex problems can be broken into easily understandable subprograms
- Subprograms can be reused in multiple applications

PL/SQL Main Features

- PL/SQL is a procedural language
- PL/SQL supports the declaration of
  - Constants
  - Variables
- Using PL/SQL, a developer can
  - Control program flow
  - Define subprograms
  - Handle runtime errors

PL/SQL Main Features

- PL/SQL provides:
  - Blocks
  - Variables and Constants
  - Error Handling
  - Subprograms
  - Packages
  - Triggers
  - Input and Output
  - Data Abstraction
  - Control Statements
  - Conditional Compilation
  - Processing a Query Result Set One Row at a Time

PL/SQL Main Features
Blocks

- The basic unit of a PL/SQL source program is the block
- The block groups related declarations and statements.
- A PL/SQL block is defined by the keywords:
  - DECLARE - declarative part
  - BEGIN - an executable part
  - EXCEPTION - an exception-handling part
  - END.
- Only the executable part is required (BEGIN ... END)
- A block can have a label.

PL/SQL Main Features
Blocks

```
<< label >> (optional)
DECLARE    -- Declarative part (optional)
  -- Declarations of local types, variables, & subprograms

BEGIN      -- Executable part (required)
  -- Statements (which can use items declared in declarative part)

[EXCEPTION -- Exception-handling part (optional)
  -- Exception handlers for exceptions (errors) raised in executable part]
END;
```

PL/SQL Main Features
Blocks

- Between DECLARE and BEGIN
  - Declaration part of the block
  - A developer can declare here:
    - Variables
    - Cursors
    - User-defined data types
    - User-defined exceptions
    - Subprograms (Functions, Procedures)
  - This part is optional

PL/SQL Main Features
Blocks

- Between BEGIN and EXCEPTION
  - Executable part of the block
  - Contains
    - Executable statements
    - SQL queries
    - Control structures
    - Loop structures
  - The EXECEPTION part of the block is optional
    - In this case, the block ends with the key-word END.

PL/SQL Main Features
Blocks

- Between EXCEPTION and END
  - Exception handler part of the block
  - This part is optional

PL/SQL Main Features
Blocks

- Block example:

```
BEGIN
    null;
END;
/
```

PL/SQL Main Features
Blocks

- There should be at least on instruction in the execution part of the block (BEGIN … END)
- Each instruction in a PL/SQL block ends with a semicolon (**;**)
- There is no semicolon after DECLARE, BEGIN and EXCEPTION
- There is a semicolon after END
- To run the block in SQL*Plus use:
  - Slash (/)
  - Run (r)

Examples

```
SET AUTOCOMMIT OFF

BEGIN
  INSERT INTO countries
    VALUES('RO', 'Romania', 1);
  INSERT INTO locations(location_id, city, country_id)
    VALUES(4000, 'Bucuresti', 'RO');
  INSERT INTO departments(department_id, department_name, location_id)
    VALUES(500, 'IT Bucuresti', 4000);
  COMMIT;
END;
/
```

Examples

```sql
SELECT * FROM countries
  WHERE country_id='RO';

SELECT * FROM locations
  WHERE location_id=4000;

SELECT * FROM departments
  WHERE department_id=500;
```

PL/SQL Main Features
Blocks

- Declarations are local to the block
- Declarations cease to exist when the block completes execution
  - This helps to avoid cluttered namespaces for variables and subprograms
- Blocks can be nested:
  - A block is an executable statement
  - A block can appear in another block wherever an executable statement is allowed.

PL/SQL Main Features
PL/SQL Nested Blocks

```
 1 [<<label>>] -- block label (optional)
 2 [DECLARE] -- optional
 3   -- variables for all the blocks
 4 BEGIN -- start of parent block
 5     [DECLARE] -- optional
 6       -- variables for child_1 block
 7     BEGIN -- start of child_1 block
 8       -- can access variables from parent and child_1 blocks
 9     [EXEPTION] -- optional
10       -- exception handler for child_1 block
11     END; -- end of child_1 block
12
13     [DECLARE] -- optional
14       -- variables for child_2 block
15     BEGIN -- start of child_2 block
16       -- can access variables from parent and child_2 blocks
17     [EXCEPTION] -- optional
18       -- exception handler for child_2 block
19     END; -- end of child_2 block
20 [EXCEPTION]  -- optional
21   -- exception handler for parent block and
22   -- any unhandled exception from the child_1 or child_2 blocks
23 END;
24 /
```

PL/SQL Main Features
PL/SQL Nested Blocks

```
[<<label>>] -- block label (optional)
[DECLARE] -- optional
  -- variables for all the blocks
BEGIN -- start of parent block

    [DECLARE] -- optional
        -- variables for child_1 block
    BEGIN -- start of child_1 block
        -- can access variables from parent and child_1 blocks

        [DECLARE] -- optional
           -- variables for child_2 block
        BEGIN -- start of child_2 block
           -- can access variables from parent, child_1 and child_2 blocks

           [DECLARE] -- optional
               -- variables for child_3 block
           BEGIN -- start of child_3 block
               -- can access variables from parent and child_3 blocks
           [EXCEPTION] -- optional
               -- exception handler for child_3 block
           END; -- end of child_3 block


        [EXCEPTION] -- optional
          -- exception handler for child_2 block
          -- handler unhandler exception from child_3 block
        END; -- end of child_2 block


    [EXEPTION] -- optional
      -- exception handler for child_1 block
      -- handler unhandler exception from child_2 and child_3 blocks
    END; -- end of child_1 block

[EXCEPTION]  -- optional
  -- exception handler for parent block and
  -- handler unhandled exception from child_1, child_2 and child_3 blocks
END;
/
```

PL/SQL Main Features
Blocks

- A block can be
  - Submitted to an interactive tool (such as SQL*Plus or Enterprise Manager)
  - Embedded in an Oracle Precompile
  - Embedded in an OCI (Oracle Call Interface) program
- The interactive tool or program runs the block one time

PL/SQL Main Features
Blocks

- The block is not stored in the database
- It is called an anonymous block (even if it has a label)
- An anonymous block is compiled each time it is loaded into memory
- An anonymous block compilation has three stages:
  1. Syntax checking: PL/SQL syntax is checked, and a parse tree is generated.
  2. Semantic checking: Type checking and further processing on the parse tree.
  3. Code generation

Examples

```
1 BEGIN
2   UPDATE locations
3     SET postal_code = '252525',
4     state_province = 'Sector 7'
5     WHERE location_id = 4000;
6   COMMIT;
7 END;
8 /
```

PL/SQL Main Features
Variables and Constants

- In the DECLARATION part of a PL/SQL block there can be declared
  - Variables
  - Constants
- They can be use them wherever you can use an expression
- As the program runs
  - The values of variables can change
  - The values of constants cannot

PL/SQL Main Features
Variables and Constants

- Example

```
DECLARE
    A NUMBER(5,2);
    PI CONSTANT NUMBER(3,2) := 3.14;
BEGIN
    A := PI * &r ** 2;
END;
/
```

PL/SQL Main Features
Error Handling

- PL/SQL supports error handlers to efficiently detect errors

- When an error occurs:
  - An exception is raised
  - Normal execution stops
  - The control transfers to the exception-handling part of the PL/SQL block.

PL/SQL Main Features Subprograms

- A PL/SQL subprogram is a named PL/SQL block that can be invoked repeatedly
- If the subprogram has parameters, their values can differ for each invocation
- PL/SQL has two types of subprograms
  1. Procedures – does not return a result
  2. Functions – returns a result
- PL/SQL also lets you invoke external programs written in other languages

PL/SQL Main Features
Packages

- A PL/SQL subprogram is a named PL/SQL block that can be invoked repeatedly

- A package is a schema object that groups logically related PL/SQL
  - Types
  - Variables
  - Constants
  - Subprograms
  - Cursors
  - Exceptions

PL/SQL Main Features
Packages

- A package is compiled and stored in the database
- Applications can share package
- Developers can write their own packages
- Oracle provides many product-specific packages

PL/SQL Main Features
Triggers

- A trigger
  - Is a named PL/SQL unit
  - Is stored in the database
  - Runs in response to an event that occurs in the database

PL/SQL Main Features
Triggers

- A developer can specify
  - The event to be handled by the trigger
  - Whether the trigger fires
    - Before the event
    - After the event
  - Whether the trigger runs for each:
    - Event
    - Row affected by the event

PL/SQL Main Features
Input and Output

- Most PL/SQL input and output (I/O) is done with SQL statements
- These statements
  - Store data in database tables
  - Query those tables
- PL/SQL I/O is also done with PL/SQL packages that Oracle Database supplies

PL/SQL Main Features
Input and Output

Package	Description	More Information
DBMS_OUTPUT	Lets PL/SQL blocks, subprograms, packages, and triggers display output. Especially useful for displaying PL/SQL debugging information.	*Oracle Database PL/SQL Packages and Types Reference*
HTF	Has hypertext functions that generate HTML tags (for example, the HTF.ANCHOR function generates the HTML anchor tag <A>).	*Oracle Database PL/SQL Packages and Types Reference*
HTP	Has hypertext procedures that generate HTML tags.	*Oracle Database PL/SQL Packages and Types Reference*
DBMS_PIPE	Lets two or more sessions in the same instance communicate.	*Oracle Database PL/SQL Packages and Types Reference*
UTL_FILE	Lets PL/SQL programs read and write operating system files.	*Oracle Database PL/SQL Packages and Types Reference*
UTL_HTTP	Lets PL/SQL programs make Hypertext Transfer Protocol (HTTP) callouts, and access data on the Internet over HTTP.	*Oracle Database PL/SQL Packages and Types Reference*
UTL_SMTP	Sends electronic mails (emails) over Simple Mail Transfer Protocol (SMTP) as specified by RFC821.	*Oracle Database PL/SQL Packages and Types Reference*

PL/SQL Main Features
Data Abstraction

- A developer can specify their own data types
- Data structures can be used to achieve data abstraction
- Developers can use:
  - Cursors
  - Composite Variables
  - The %ROWTYPE Attribute
  - The %TYPE Attribute
  - Abstract Data Types

PL/SQL Main Features
Data Abstraction

- A cursor
  - Is a pointer to a private SQL area that stores information about processing
    - a specific SQL statement
    - PL/SQL SELECT INTO statement
  - Is used to retrieve the rows of the result set one at a time.
- Cursor attributes can be used to get information about the state of the cursor (e.g., how many rows the statement has affected so far)

PL/SQL Main Features
Data Abstraction

- A composite variable has internal components
- An internal component can be accessed individually
- Entire composite variables can be passed to subprograms as parameters

PL/SQL Main Features
Data Abstraction

- PL/SQL has two kinds of composite variables:
  - Collections
    - The internal components (elements) are always of the same data type
    - Each element is accessed by its unique index.
    - E.g., lists and arrays
  - Records
    - The internal components (fields) can be of different data types
    - Each field can be accessed by its name
    - A record variable can hold
      - A table row
      - Some columns from a table row.

PL/SQL Main Features
Data Abstraction

- The %ROWTYPE attribute
  - Is used to declare a record that represents either
    - A full row of a database table or view
    - Partial row of a database table or view

- The %TYPE attribute
  - Is used to declare a data item of the same data type as a previously declared variable or column

Examples

```
SET SERVEROUTPUT ON

DECLARE
  dn departments.department_name%TYPE;
  idd departments.department_id%TYPE := 500;
BEGIN
  SELECT department_name INTO dn
    FROM departments
    WHERE department_id = idd;
  DBMS_OUTPUT.PUT_LINE('Department id:   ' || idd);
  DBMS_OUTPUT.PUT_LINE('Department name: ' || dn);
END;
/
```

Examples

```
SET SERVEROUTPUT ON

DECLARE
  dn departments.department_name%TYPE;
  idd departments.department_id%TYPE := &id;
BEGIN
  SELECT department_name INTO dn
    FROM departments
    WHERE department_id = idd;
  DBMS_OUTPUT.PUT_LINE('Department id:   ' || idd);
  DBMS_OUTPUT.PUT_LINE('Department name: ' || dn);
END;
/
```

Examples

```
SET SERVEROUTPUT ON

DECLARE
  dn departments.department_name%TYPE;
BEGIN
  SELECT department_name INTO dn
    FROM departments
    WHERE department_id = &&idd;
  DBMS_OUTPUT.PUT_LINE('Department id:   '||&idd);
  DBMS_OUTPUT.PUT_LINE('Department name: '||dn);
END;
/

define
undef idd
```

Examples

```
SET SERVEROUTPUT ON

DECLARE
  country countries%ROWTYPE;
BEGIN
  SELECT * INTO country
    FROM countries
    WHERE country_id = &id;
  DBMS_OUTPUT.PUT_LINE('Country id:   '|| country.country_id);
  DBMS_OUTPUT.PUT_LINE('Country name: '|| country.country_name);
END;
/
```

PL/SQL Main Features
Data Abstraction

- An Abstract Data Type (ADT) consists of a data structure and subprograms that manipulate the data
- The variables that form the data structure are called attributes
- The subprograms that manipulate the attributes are called methods
- ADTs are stored in the database
- Instances of ADTs can be
  - Stored in tables
  - Used as PL/SQL variables.

PL/SQL Main Features
Data Abstraction

- ADTs reduce complexity by separating a large system into logical reusable components
- ADTs are created using CREATE TYPE Statement.
- ADTs are also called user-defined types and object types

PL/SQL Main Features
Conditional Compilation

- Conditional compilation enables developers to customize the functionality in a PL/SQL application without removing source text

- For example:
  - Use new features with the latest database release
  - Disable new features when running the application in an older database release
  - Activate debugging or tracing statements in the development environment
  - Hide debugging or tracing statements when running the application at a production site

PL/SQL Main Features

Processing a Query Result Set One Row at a Time

- Inside PL/SQL, the results of a SQL query can be processed one row a time
- This can be achieved by using:
  - A basic loop
  - An individual statements to
    - Run the query
    - Retrieve the results
    - Finish processing

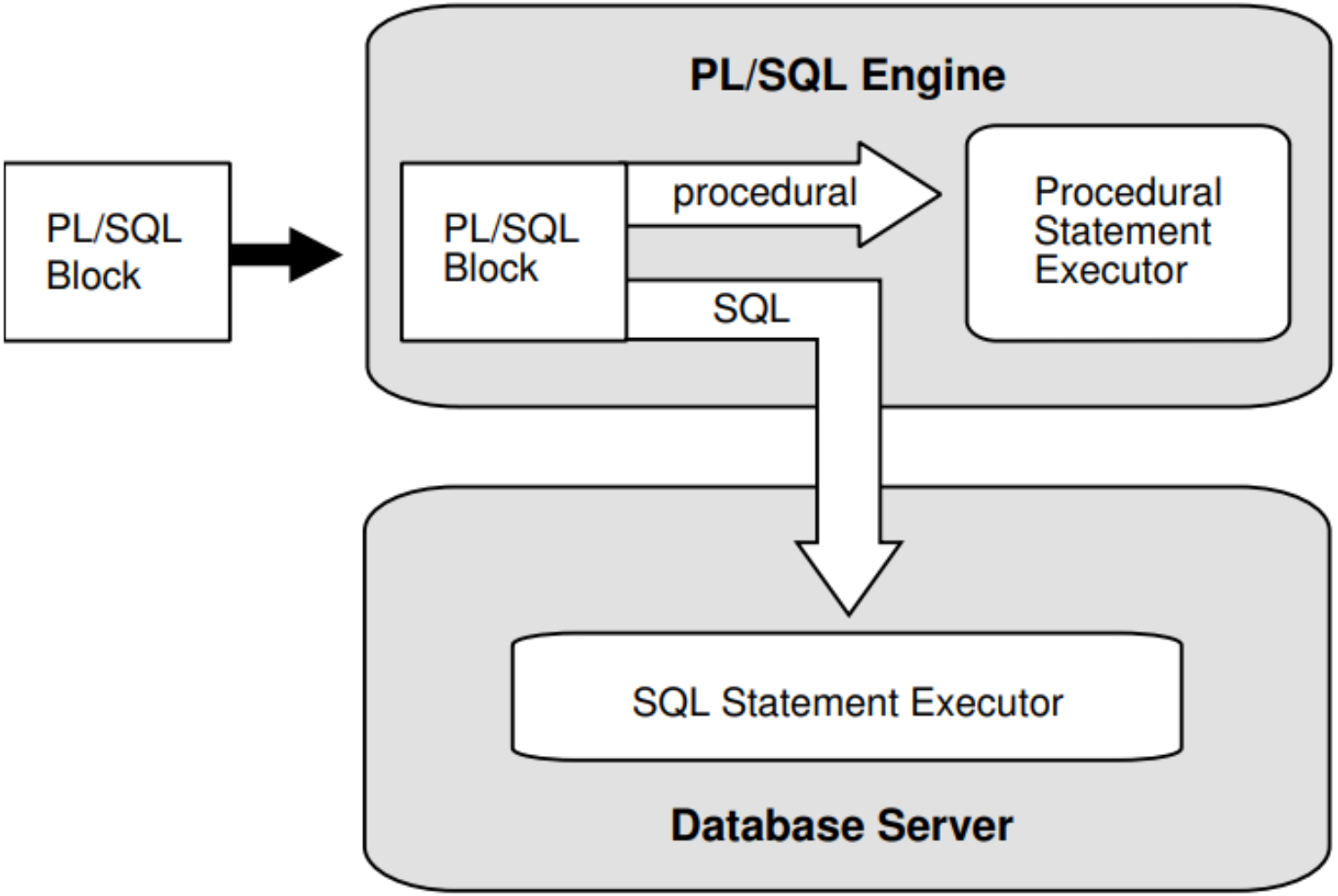
Overview

- PL/SQL
- PL/SQL Main Features
- PL/SQL Engine

PL/SQL Engine

- The PL/SQL compilation and runtime system is an engine
- The PL/SQL engine compiles and runs PL/SQL units
- The PL/SQL engine can be installed:
  - In the database
  - In an application development tool (e.g., Oracle Forms)
- Regardless of the environment, the PL/SQL engine accepts as input any valid PL/SQL unit

PL/SQL Engine



PL/SQL Engine

- The engine
  - Runs procedural statements
  - Sends SQL statements to the SQL engine in the database
  - PL/SQL units are processed by:
    - The database if the database PL/SQL engine is used
    - The local PL/SQL engine if an application development tool is used
  - If a PL/SQL unit contains no SQL statements
    - The local engine processes the entire PL/SQL unit
    - This is useful if the application development tool can benefit from conditional and iterative control

PL/SQL Engine

- Why using PL/SQL Engine is useful:
  - Oracle Forms applications frequently use SQL statements to
    - Test the values of field entries
    - Do simple computations
  - By using PL/SQL instead of SQL
    - Avoid calls to the database

Examples

```
 1 DECLARE
 2   country countries%ROWTYPE;
 3   region regions%ROWTYPE;
 4 BEGIN
 5   SELECT
 6     c.country_id,
 7     c.country_name,
 8     r.region_id,
 9     r.region_name
10   INTO
11     country.country_id,
12     country.country_name,
13     region.region_id,
14     region.region_name
15   FROM countries c
16     INNER JOIN regions r
17       on c.region_id = r.region_id
18     WHERE country_id = &id;
19   DBMS_OUTPUT.PUT_LINE('Country id:   '|| country.country_id);
20   DBMS_OUTPUT.PUT_LINE('Country name: '|| country.country_name);
21   DBMS_OUTPUT.PUT_LINE('Region id:    '|| region.region_id);
22   DBMS_OUTPUT.PUT_LINE('Region name:  '|| region.region_name);
23 END;
24 /
```

Examples

```
1 BEGIN
2   DELETE FROM departments
3     WHERE department_id = 500;
4   DELETE FROM locations
5     WHERE location_id = 4000;
6   DELETE FROM countries
7     WHERE country_id = 'RO';
8   COMMIT;
9 END;
10 /
11
```

Bibliography

- Usha Krishnamurthy et al. *Oracle® Database: SQL Language Reference 19c*, Oracle Corporation, 2022 [pdf]
- Usha Krishnamurthy et al. *Oracle® Database: SQL Language Reference 21c*, Oracle Corporation, 2022 [pdf]
- Usha Krishnamurthy et al. *Oracle® Database: SQL Language Reference 23ai*. Oracle Corporation 2024 [pdf]
- Louise Morin et al. *Oracle® Database: Database PL/SQL Language Reference 19c*, Oracle Corporation, 2020 [pdf]
- Louise Morin et al. *Oracle® Database: Database PL/SQL Language Reference 21c*, Oracle Corporation, 2021 [pdf]
- Sarah Hirschfeld et al. *Oracle® Database: Database PL/SQL Language Reference 23ai*. Oracle Corporation 2024 [pdf]