SQL mastered

Revision of PL/SQL

What is PL/SQL?

- As SQL is a non-procedural language and often we need to put structure on transactions, so we need PL/SQL to provide the supplemental structures:
  - For sequence, selection, iteration
  - Variables, constants and data types
  - Assignment and other arithmetic statements
  - Customised error handling

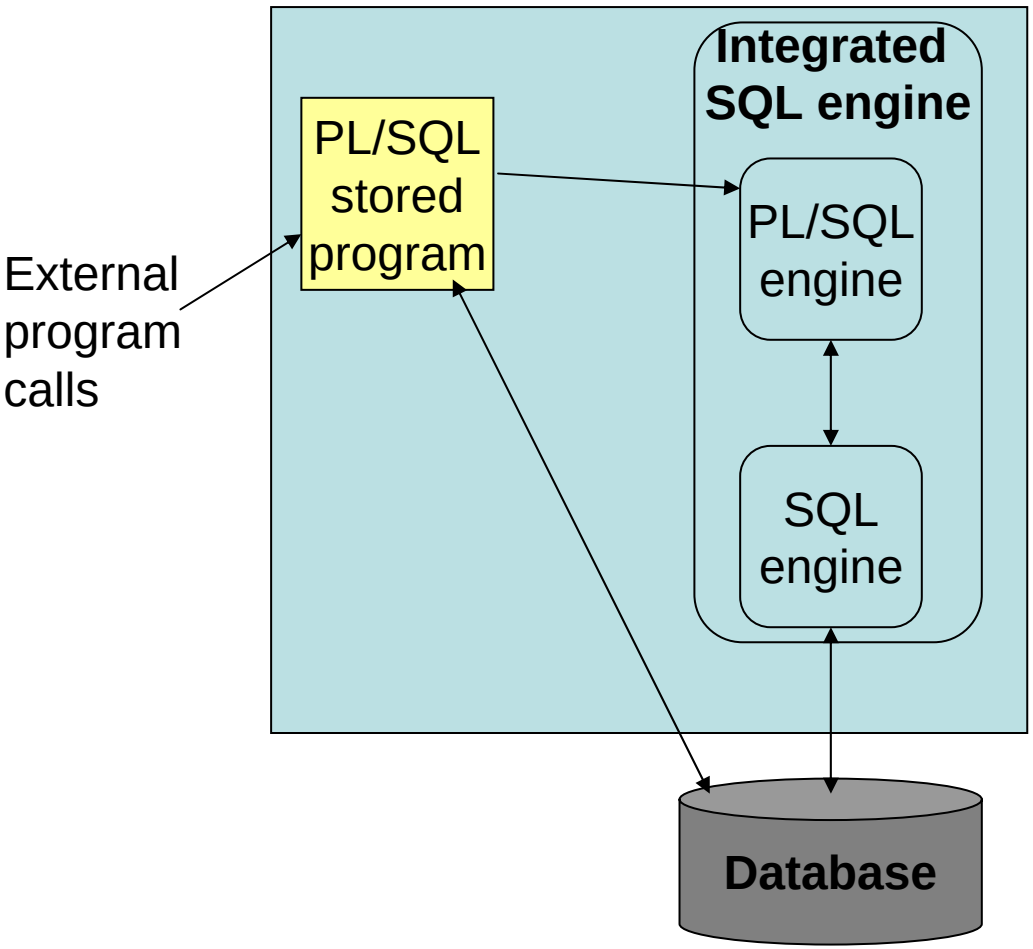
What is PL/SQL?

- A superset of the Data Manipulation Language part of SQL.
- It allows the user to store compiled code in the database, giving central access to routines associated with the data.

How is PL/SQL handled?

- SQL is interpreted by an SQL engine. This is functionality that is available in all databases.
- There is also a PL/SQL engine that interprets or executes compiled PL/SQL code. It extracts any SQL and passes it on to the SQL engine.

PL/SQL architecture



PL/SQL architecture explained

- When a call is made from a program to the server to run a PL/SQL program
  - The Oracle database loads the compiled program into memory
  - The PL/SQL engine handles the program's memory structures and logical program flow.
  - Any SQL queries in the program are handed on to the SQL engine to issue data requests.

Simplest version

- Add BEGIN to the start of your transaction
- Add END to the end.
- This is now an anonymous PL/SQL block.
- The anonymous block is the basis for all programs.
- It is made up of up to three sections:
  - Declarations
  - Executable section
  - Exception handling section

The Executable section

- The ability to run queries is supplemented by the ability to include logical structures.
- PL/SQL allows for sequence, selection and iteration.
- It allows for nesting, parameter passing.
- It allows the embedding of functions in objects (not relational).

The basic structure

```
[DECLARE]
-- Put variables here.
BEGIN
-- Put program here.
[EXCEPTION]
-- Put exception handlers here.
END;
```

Anonymous block structure

```
Declare
  -- Declaration Section
  -- Data, subprogram declarations
begin
  -- Executable Section
  null; -- Program statements
Exception
  -- Exception section
  -- Exception handlers
  when others then
     null; -- default handler
end;
```

Declarations

- The data types in PL/SQL
  - The most common re consistent with SQL datatypes and are
    - Varchar2
    - Number
    - Date
    - Boolean
- Data Definition Language aspect
  - PL/SQL does not provide for the Data Definition Language aspect of SQL.
  - There is a package called DBMS_SQL that allows for data definition statements.

Scalar Data types

- Numeric
  - **binary_integer integer from -2^{31} -1 to $2^{31} - 1$**
  - **natural integer from 0 to 2^{31}**
  - **positive integer from 1 to 2^{31}**
  - Number(p,s) p is precision, s is scale
- Character
  - char(n) fixed length string of length n
  - varchar2(n) variable length string of maximum length n
- Other
  - **Boolean boolean data type (true, false)**
  - Date, time date Same as Oracle's date.

Variable Declaration

- <variable-name> <datatype> [not null] [:=<initial-value>];
- <constant-name> constant <datatype> := <value>
- E.g.

```
i BINARY_INTEGER;
custno NUMBER(5) NOT NULL:= 1111;
Custname VARCHAR2(30);
Unit_price NUMBER(5,2):=12.5;
Percent_VAT CONSTANT INTEGER(2):=20;
Order_date DATE;
Delivered BOOLEAN;
```

Look at the list

- i is an integer, with no initial or default value and no constraints.
- custno is a 5 digit number that cannot be NULL and has an initial value of 1111.
- custname is a string up to 30 characters long, with no initial or default value and no constraints.
- Unit_price is a decimal number that can contain a value up to 999.99 and has an initial value of 12.5.
- Order_date is a date and must conform to date constraints.
- Delivered can be TRUE, FALSE or NULL. It has no initial value.

Comments

- Comments can either be surrounded by /*..*/ or can be preceded, in any part of the middle of a line, by --
- E.g.

```
i  binary_integer :=0; --counter
```

More advanced datatypes

- Rather than declaring a variable completely independently of stored data, the variable can be declared to be ***of the same type as*** a column in a table:

```
temp_custno customer.custno%TYPE;
```

- This removes the need for the programmer to constantly watch the datatypes of columns in stored tables.
  - If the procedure is to be run from outside the schema, then the full pathname of the column is required:

```
Stockvar builder2.stock.stock_code%type
```

Executable section

- This holds our program code.
- If an error occurs in the program code, then, if there is an exception section, control will go to that section.
  - NB: Once control goes to the exception section, it does not return. This is a branch statement, not a CALL.

Exceptions

- Oracle defined exceptions.
  - Relate to violations of the expected outcome of a statement, as defined by Oracle. E.g. insert duplicate index
- User defined exceptions.
  - These are user-written exceptions, that can be incorporated into PL/SQL routines.
  - Examples:
    - When we allocate stock to a customer order, if the stock falls below reorder level, we want to log the fact that the stock needs to be reordered, but continue with the transaction.
    - When we try to allocate stock to a customer order, but there isn’t enough stock, we need to have a high level emergency warning for reordering.

I/O to a PL/SQL block

- PL/SQL conducts I/O to stored data through SQL.
- In order to communicate to the developer, there is a package that allows the PL/SQL block to output to the session buffer.
- This package is called DBMS_OUTPUT.
- It contains the following functions:
  - PUT_LINE
  - PUT
  - NEW_LINE

Using the session buffer

- The data displayed by the DBMS_OUTPUT package can only be seen if the session's buffer is on.
- To turn on the session buffer display, in SQL*Plus or iSQLPlus, use the command:
- SET SERVEROUTPUT ON
- This is a session command – it is not part of PL/SQL and should not go into the PL/SQL block.

Take an example (Builder2)

- Take in a stock code through a substitution variable.
- Select the stock description from the stock table in the builder2 schema that matches the supplied stock code.
- Display the message “Stock item <stock_code> is a <stock_description>”.

Code

```
DECLARE
scod builder2.stock.stock_code%TYPE:=&scod;
sdesc builder2.stock.stock_description%TYPE;
BEGIN
   SELECT stock_description INTO sdesc
   FROM builder2.stock
   WHERE stock_code = scod;
   DBMS_OUTPUT.PUT_LINE('Stock item
   '||scod||' is a '||sdesc||'.');
END;
```

- *Can you remember what the items circled in red are?*

Control structures

- PL/SQL allows for
  - Sequence
  - Selection
    - IF-THEN – END IF;
    - IF – THEN – ELSE – END IF;
  - Iteration

selection If-then statement:

```
IF <condition> THEN
 <stmt>;
END IF;
```

Example:

```
IF(quantityrequired > stock_level) THEN
  DBMS_OUTPUT.PUT_LINE('Stock level is only'
                        ||Stock_level);
END IF;
```

N.B. The apostrophes '' in PowerPoint are different to those in SQL or Notepad.

If-then-else statement:

```
IF(quantityrequired > stock_level) THEN
  DBMS_OUTPUT.PUT_LINE('Stock level is only '
                        ||Stock_level);
ELSE
  DBMS_OUTPUT.PUT_LINE('Now have '
          ||(stock_level-quantityrequired)
          ||' items remaining');
  stock_level:= stock_level–quantityrequired;
END IF;
```

The exception section

- There are a certain number of exceptions that are handled by Oracle.
  - Some of these are listed on the following slide.
- There are other exceptions that you can include.
- In the exception section, exceptions are listed, something like a case statement.
- When an exception occurs, control goes to the exception section, if one exists.
- It tests each exception handler to see if it handles the exception that has occurred. If so, it uses that exception handler and exits.
  - The ‘others’ handler is a catch-all.

Named system exceptions

- Oracle can handle:
  - **CURSOR_ALREADY_OPENED**
  - **DUP_VAL_ON_INDEX**
  - **INVALID_CURSOR**
  - **INVALID_NUMBER**
  - **LOGIN_DENIED**
  - **NO_DATA_FOUND**
  - **TOO_MANY_ROWS**
- These are named in the ‘standard’ package in pl/sql.

What if there’s no such code?

- If there’s an error in data entry, then the SELECT statement will fail.
- There are certain errors that are listed as exceptions. These include:
  - DUP_VAL_ON_INDEX
  - NO_DATA_FOUND
  - TOO_MANY_ROWS
- These can be handled in the EXCEPTION section.

The Exceptions Section

- This section is for handling errors.
- It contains one or more exception handlers.
- When an error occurs, control is passed to this section.
- Each exception handler is checked sequentially until the exception that occurred is reached.
- If the exception that occurred is reached, then the handler is invoked.
  - **Otherwise, the block finishes, leaving the exception active.**
    - **This is sometimes what you want.**

Exception handlers

- Say we believe that it is likely that a user may request a select using a condition that will yield no results.
- We are expecting there to be no data to satisfy that query- NO_DATA_FOUND
- The exception handler for this exception will be:

```
WHEN NO_DATA_FOUND THEN
…code
```

WHEN OTHERS THEN

- There is a catchall exception handler that will handle any exception that is not explicitly listed.
- When used, it is listed at the very end of the list of exceptions.
- If the transaction is half-way through, then we may want to cancel it – i.e. rollback the work that has been done.

Exceptions example

```
EXCEPTION
WHEN NO_DATA_FOUND THEN
  DBMS_OUTPUT.PUT_LINE (‘No data was
  found’);
WHEN DUP_VAL_ON_INDEX THEN
  DBMS_OUTPUT.PUT_LINE(‘Record already
  there);
WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE(‘An unexpected error
  has occurred’);
  ROLLBACK WORK;
END;
```

What about the stock code?

```
--
-- Don't forget to SET SERVEROUTPUT ON before you run this.
--
DECLARE
  scod builder2.stock.stock_code%TYPE:=&scod;
  sdesc builder2.stock.stock_description%TYPE;
BEGIN
  SELECT stock_description INTO sdesc
  FROM builder2.stock
  WHERE stock_code = scod;
  DBMS_OUTPUT.PUT_LINE('Stock item '||scod||' is a
EXCEPTION
WHEN NO_DATA_FOUND THEN
  DBMS_OUTPUT.PUT_LINE('Stock item '||scod||' does not exist.');
WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE('An unexpected error has occurred.');
END;
```

Explanation

- If the stock code is not retrieved from the stock table, then the first error is displayed.
- If any other error occurs, the second error message is displayed.

Using the session buffer

- The data displayed by the DBMS_OUTPUT package can only be seen if the session's buffer is on.
- To turn on the session buffer display, in SQL*Plus or iSQLPlus, use the command: SET SERVEROUTPUT ON
- This is a session command – it is not part of PL/SQL and should not go into the PL/SQL block.

Another example

```
DECLARE
cust_no BUILDER2.customer.customer_id%type :=
&cust_no;
cust_name BUILDER2.customer.customer_name%type;
cust_addr BUILDER2.customer.customer_address%type;
BEGIN
  SELECT customer_name, customer_address
  INTO cust_name, cust_addr
  FROM builder2.customer
  WHERE customer.customer_id = cust_no;
  dbms_output.put_line(cust_no||' '||cust_name
                       ||' '||cust_addr);
EXCEPTION
WHEN NO_DATA_FOUND THEN
  dbms_output.put_line('Customer ' || cust_no ||
                       ' not found');
WHEN OTHERS THEN
dbms_output.put_line(
            'An undefined error has occurred');
END;
```