BIT360– Advanced Databases

- CA – 40%
  - 2 TESTS
- EXAM – 60%
- www.LeChaamwe.weebly.com
  - Lecture Notes
  - Undergraduate

Reading Materials

- **Prescribed Textbooks**
- Thomas Connolly, Carolyn Begg "Database Systems A Practical Approach to Design, Implementation, and Management" Sixth Edition (2015)
- **Recommended Textbooks**
- Elmasri, Navathe, Somayajulu, and Gupta, "Fundamentals of Database Systems", 7th Edition, Pearson Education (2016)
- Silberschatz A., Korth H.F. and Sudarshan S., "Database System Concepts", 6th Edition, McGraw Hill (2010).

Procedures

Stored Procedures (Intro)

- Data retrieval has in most cases been accomplished with a single statement.
- Even the use of subqueries is accomplished by combining two or more SELECTs into a single statement.
- multiple statements can be saved into a single object known as a **stored procedure**.

Stored Procedures

- A **Stored Procedure** is a set of SQL statements, compiled and stored as a single database object *for repeated use.*
- It is used to get information from the database or change data in the database
  - It is used by **application** programs (along with **views**)

Stored Procedures

- It can use **zero** or **more** **parameters**
- It is run using an **EXECUTE** statement (in MS SQL SERVER) or **CALL** (in MySQL) with the procedure name and any parameter values
- It is built using a CREATE PROCEDURE statement.

Stored Procedures

- there are two general reasons why we might want to use stored procedures:
  - To save **multiple** SQL statements in a **single** procedure
  - To use **parameters** in conjunction with your SQL statements
- Stored procedures can, in fact, consist of a single SQL statement and contain no parameters.

Stored Procedures

- ***But the real value of stored procedures becomes evident when they contain multiple statements or parameters.***
- This is something that relates directly to the issue of how to best retrieve data from a database.

Stored Procedures

- **The ability to store multiple statements in a procedure means that you can create complex logic and execute it all at once as a single transaction.**
- For example, you might have a business requirement to take an incoming order from a customer and quickly evaluate it before accepting it from the customer.

Stored Procedures

- This procedure *might* involve:
  - **checking** to make sure that the items are in **stock**
  - **verifying** that the customer has a good **credit rating**
  - **getting an initial estimate** as to when it can be **shipped**

Stored Procedures

- This situation would require multiple SQL statements with some added logic to determine what kind of message to return if all were not well with the order.
- All of that logic could be placed into <u>a single stored procedure</u>, which would enhance the modularity of the system.

Stored Procedures

- With everything in one procedure, that logic could be executed from any calling program, and it would always return the same result.
- Stored procedures can be called from:
  - Programs written in standard languages, e.g., Java, C#
  - Scripting languages, e.g., JavaScript, VBScript
  - SQL command prompt, e.g., SQL*Plus, Query Analyzer

Benefits of Stored Procedures

- **Modular Programming** – You can write a stored procedure once, then call it from multiple places in your application.
- **Performance** - Stored procedures provide faster code execution and reduce network traffic.

Benefits of Stored Procedures

- This means that it will execute a lot faster than sending many lines of SQL code from your application to the SQL Server.
- Doing that requires SQL Server to compile and optimze your SQL code every time it runs.

Benefits of Stored Procedures

- Reduced network traffic: If you send many lines of SQL code over the network to your SQL Server, this will impact on network performance.
- This is especially true if you have hundreds of lines of SQL code and/or you have lots of activity on your application.

Benefits of Stored Procedures

- Running the code on the SQL Server (as a stored procedure) eliminates the need to send this code over the network.
- The only network traffic will be the parameters supplied and the results of any query.

Benefits of Stored Procedures

- **Security** - Users can execute a stored procedure without needing to execute any of the statements directly.
- Greater security as store procedures are always stored on the database server

Disadvantages of Stored Procedures

- Increased **load** on the database server — most of the work is done on the **server side**, and less on the client side.
- You'll need to learn not only the syntax of SQL statements in order to write stored procedures, but the particular "**dialect**" of the **DBMS** managing them (e.g., SSQL Server T-SQL vs. MySQL vs Oracle vs DBs)

Disadvantages of Stored Procedures

- **Migrating** to a different **database management system** (MySQL, SQL Server, Oracle, DB2, etc) may potentially be more difficult

Procedures in Oracle

- Oracle uses a Language known as PL/SQL to implement Procedures

PL/SQL

- PL/SQL stands for Procedural Language extension of SQL.
  PL/SQL is a combination of SQL along with the procedural features of programming languages.
- It was developed by Oracle Corporation in the early 90's to enhance the capabilities of SQL.

The PL/SQL Engine:

- Oracle uses a PL/SQL engine to processes the PL/SQL statements.
- A PL/SQL code can be stored in the client system (client-side) or in the database (server-side).

A PL/SQL Block:

- Each PL/SQL program consists of SQL and PL/SQL statements
- which form a PL/SQL block.
- A PL/SQL Block consists of three sections:
  - The Declaration section (optional).
  - The Execution section (mandatory).
  - The Exception (or Error) Handling section (optional).

Declaration Section

- The Declaration section of a PL/SQL Block starts with the reserved keyword DECLARE.
- This section is optional and is used to declare any placeholders like variables, constants, records and cursors,

Declaration Section

- which are used to manipulate data in the execution section.
- Placeholders may be any of Variables, Constants and Records, which stores data temporarily.
- Cursors are also declared in this section.

Execution Section

- The Execution section of a PL/SQL Block starts with the reserved keyword BEGIN and ends with END.
- This is a mandatory section and is the section where the program logic is written to perform any task.
- The programmatic constructs like loops, conditional statement and SQL statements form the part of execution section.

Exception Section

- The Exception section of a PL/SQL Block starts with the reserved keyword EXCEPTION.
- This section is optional.
- Any errors in the program can be handled in this section,
- so that the PL/SQL Blocks terminates gracefully.

Exception Section

- If the PL/SQL Block contains exceptions that cannot be handled, the Block terminates abruptly with errors.
- Every statement in the above three sections must end with a semicolon ; .
- PL/SQL blocks can be nested within other PL/SQL blocks.
- Comments can be used to document code.

PL/SQL Block

- This is how a sample PL/SQL Block looks.

```
DECLARE
   Variable declaration
BEGIN
   Program Execution
EXCEPTION
   Exception handling
END;
```

PL/SQL Placeholders

- Placeholders are temporary storage areas.
- Placeholders can be any of Variables, Constants and Records.
- Oracle defines placeholders to store data temporarily,
- which are used to manipulate data during the execution of a PL SQL block.

PL/SQL Placeholders

- Depending on the kind of data you want to store,
- you can define placeholders with a name and a datatype.
- Few of the datatypes used to define placeholders are as given below.
- Number (n,m) , Char (n) , Varchar2 (n) , Date , Long , Long raw, Raw, Blob, Clob, Nclob, Bfile

PL/SQL Variables

- These are placeholders that store the values that can change through the PL/SQL Block.
- The General Syntax to declare a variable is:
  - variable_name datatype [NOT NULL := value];
- *variable_name* is the name of the variable.
- *datatype* is a valid PL/SQL datatype.

PL/SQL Variables

- NOT NULL is an optional specification on the variable.
- *value* or DEFAULT *value* is also an optional specification,
- where you can initialize a variable.
- Each variable declaration is a separate statement and must be terminated by a semicolon.

PL/SQL Variables

- For example,
- if you want to store the current salary of an employee,
- you can use a variable.
- DECLARE salary number (6);
- * "salary" is a variable of datatype number and of length 6.

PL/SQL Variables

- When a variable is specified as NOT NULL,
- you must initialize the variable when it is declared.
- For example: The below example declares two variables, one of which is a not null.
- DECLARE
  - salary number(4);
  - dept varchar2(10) NOT NULL := "HR Dept";

PL/SQL Variables

- The value of a variable can change in the execution or exception section of the PL/SQL Block.
- We can assign values to variables in two ways.
- We can directly assign values to variables.
  - The General Syntax is:
  - variable_name:= value;

PL/SQL Variables

- We can assign values to variables directly from the database columns by using a SELECT.. INTO statement.
- The General Syntax is:
  - SELECT column_name INTO variable_name FROM table_name [WHERE condition];

Example

- The below program will get the salary of an employee with id '1116' and display it on the screen.
- DECLARE
  - var_salary number(6);
  - var_emp_id number(6) = 1116;
- BEGIN SELECT salary INTO var_salary
- FROM employee

Example

- WHERE emp_id = var_emp_id;
- dbms_output.put_line(var_salary);
- dbms_output.put_line('The employee ' || var_emp_id || ' has salary ' || var_salary); END;

PL/SQL Constants

- As the name implies a *constant* is a value used in a PL/SQL Block that remains unchanged throughout the program.
- A constant is a user-defined literal value.
- You can declare a constant and use it instead of actual value.

PL/SQL Constants

- For example:
- If you want to write a program which will increase the salary of the employees by 25%,
- you can declare a constant and use it throughout the program.
- Next time when you want to increase the salary again you can change the value of the constant which will be easier than changing the actual value throughout the program.

PL/SQL Constants

- The General Syntax to declare a constant is:
- constant_name CONSTANT datatype := VALUE;
- *constant_name* is the name of the constant i.e. similar to a variable name.

PL/SQL Constants

- The word *CONSTANT* is a reserved word and ensures that the value does not change.
- *VALUE* - It is a value which must be assigned to a constant when it is declared.
- You cannot assign a value later.
- For example, to declare salary_increase, you can write code as follows:

PL/SQL Constants

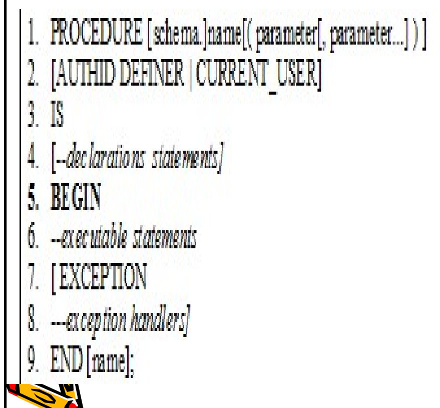
- DECLARE
- salary_increase CONSTANT number (3) := 10;
- You *must* assign a value to a constant at the time you declare it.
- If you do not assign a value to a constant while declaring it and try to assign a value in the execution section, you will get a error.

PL/SQL Constants

- If you execute the below Pl/SQL block you will get error.
- DECLARE
- salary_increase CONSTANT number(3);
- BEGIN
  - salary_increase := 100;
  - dbms_output.put_line (salary_increase);
- END;

PL/SQL procedure (Oracle)

- a ***PL/SQL procedure*** is a named block that performs one or more actions.
- PL/SQL procedure allows you to wrap complex business logic and reuse it.
- Generally, you use a procedure to perform an action and a function to compute a value.
- The following illustrates the PL/SQL procedure's syntax:



```
PROCEDURE [schema.]name[( parameter[, parameter...] ) ]
[AUTHID DEFINER | CURRENT_USER]
IS
[--declarations statements]
BEGIN
--executable statements
[EXCEPTION
---exception handlers]
END [name];
```

PL/SQL procedure (Oracle)

- We can divide the PL/SQL procedure into two sections: header and body.
- **PL/SQL Procedure's Header**
- The section before the keyword IS is called procedures' header or procedure's signature.
- The elements in the procedure's header are listed as follows:

PL/SQL procedure (Oracle)

- Schema:
  - The optional name of the schema that own this procedure.
  - The default is the current user.
  - If you specify a different user, the current user must have privileges to create a procedure in that schema.

PL/SQL procedure (Oracle)

- Name:
  - The name of the procedure.
  - The name of the procedure should be always meaningful and starting by a verb.

PL/SQL procedure (Oracle)

- Parameters:
  - The optional list of parameters.
- {parameter_name} is the name of parameter being passed to procedure along with parameter's data type {parameter_data_type}.

PL/SQL procedure (Oracle)

- AUTHID:
  - The optional AUHTID determines whether the procedure will execute with the privileges of the owner (DEFINER) of the procedure or the current user (CURRENT_USER).

PL/SQL Procedure's Body

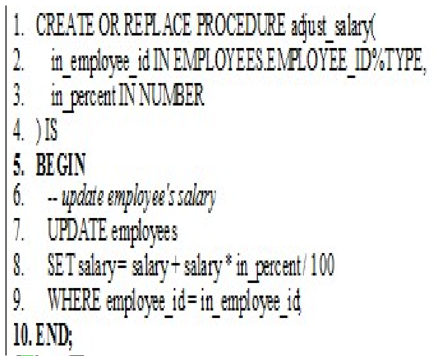
- Everything after the keyword IS is known as procedure's body.
- The procedure's body consists of declaration, execution and exception sections.
- The declaration and exception sections are optional.
- You must have at least one executable statement in the execution section.

PL/SQL Procedure's Body

- In PL/SQL procedure you have RETURN statement.
- RETURN statement in procedure is used only to halt the execution of procedure and return control to the caller.
- RETURN statement in procedure does not take any expression or constant.

Example of PL/SQL Procedures

- We're going to develop a procedure called *adjust_salary()*.
- We'll update the salary information of employees in the table *employees* by using SQL UPDATE statement.
- Here is the PL/SQL procedure *adjust_salary()* code sample:



```
CREATE OR REPLACE PROCEDURE adjust_salary(
  in_employee_id IN EMPLOYEES.EMPLOYEE_ID%TYPE,
  in_percent IN NUMBER
) IS
BEGIN
   -- update employee's salary
   UPDATE employees
   SET salary = salary + salary * in_percent / 100
   WHERE employee_id = in_employee_id;
END;
```

PL/SQL Procedure's Body

- There are two parameters of the procedure IN_EMPLOYEE_ID and IN_PERCENT.
- This procedure will update salary information by a given percentage (IN_PERCENT) for a given employee specified by IN_EMPLOYEE_ID.

PL/SQL Procedure's Body

- In the procedure's body, we use SQL UPDATE statement to update salary information.
- Let's take a look how to call this procedure.

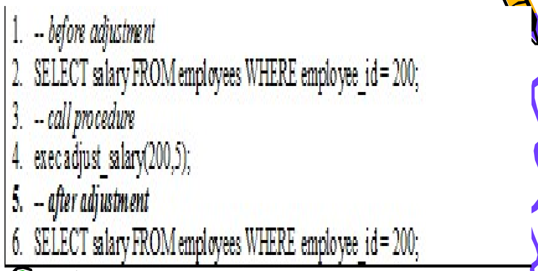
Calling PL/SQL Procedures

- A procedure can call other procedures.
- A procedure without parameters can be called directly by using keyword EXEC or EXECUTE followed by procedure's name as below:
  - EXEC procedure_name();
  - EXEC procedure_name;

Calling PL/SQL Procedures

- Procedure with parameters can be called by using keyword EXEC or EXECUTE followed by procedure's name and parameter list in the order corresponding to the parameters list in procedure's signature.
  - EXEC procedure_name(param1,param2…paramN);

Calling PL/SQL Procedures



```
-- before adjustment
SELECT salary FROM employees WHERE employee_id = 200;
-- call procedure
exec adjust_salary(200,5);
-- after adjustment
SELECT salary FROM employees WHERE employee_id = 200;
```

Questions