**The British College**

**KATHMANDU**

**Coursework Submission Coversheet**(individual coursework only)

**Faculty of Arts, Environment and Technology LBU Student Id:**

77178367

**For checking by the student:**

Please ensure all information is complete and correct and attach this form securely to the front of your work before posting it in a coursework collection box.

Award name: BSc (Hons) Computing

Module code: 23269 – 201516

Module name: Advanced Database B

Module run: 2017

Coursework title: 2

Due Date: 26 MARCH (2017)

Module leader: (In LBU) Jackie Campbell, Sanela Lazarevski

Module tutor: (In TBC) Dibya Tara Shakya

**TURNITIN** Checked: YES NO ***(please circle)***

Submission date& time: Date: 26 March (2017) Time: 10 PM

**Total Word Count: Total Number of Pages (including this front sheet):**

**In submitting this form with your assignment, you make the following declaration:**  
I declare, that the coursework submitted is my own work and has not (either in whole or part) been submitted towards the award of any other qualification either at LBU or elsewhere. I have fully attributed/referenced all sources of information used during the completion of my assignment, and I am aware that failure to do so constitutes an assessment offence.

Signed: Date:

**You are strongly advised to retain a second copy of your work in case of any query about the assignment.**

3473

23

**For completion by the faculty:**

**This mark is provisional and subject to moderation and approval by the relevant examining board**

**Teacher's Feedback**

**Teacher's Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Contents

[Course Work 2 2](#_Toc478327463)

[Introduction 2](#_Toc478327464)

[Reusable Code 2](#_Toc478327465)

[Function and Procedure 2](#_Toc478327466)

[Implementation Method 3](#_Toc478327467)

[Package 3](#_Toc478327468)

[Implementation 3](#_Toc478327469)

[Package Specification 3](#_Toc478327470)

[Package Body 3](#_Toc478327471)

[Function to GETMAXJOINDATE (Id IN NUMBER) 3](#_Toc478327472)

[Function GETSKILLNMAE (SKILLID IN NUMBER) 5](#_Toc478327473)

[Procedure PROC\_MY\_PLACEMENT 6](#_Toc478327474)

[Procedure SKILAUDIT 8](#_Toc478327475)

[Testing 8](#_Toc478327476)

[Test Cases 9](#_Toc478327477)

[Test Log 10](#_Toc478327478)

[Test Log Table 14](#_Toc478327479)

[Course Work 3 16](#_Toc478327480)

[Introduction 16](#_Toc478327481)

[Growing data for demonstration 16](#_Toc478327482)

[Performance Tuning Strategies 16](#_Toc478327483)

[Execution Plan 17](#_Toc478327484)

[Recommendation 20](#_Toc478327485)

[Export/Import Dump File 20](#_Toc478327486)

[Summary 21](#_Toc478327487)

# Course Work 2

# Introduction

This document is continuation of database application produced for PlaceU recruitment agency. Previously, security plan for PlaceU were planned, implemented and test. This document plans and implemented reusable component such as function and procedure and implement and justify performance tuning strategy for the application.

PlaceU application has two modules developed. ‘Admin’ and ‘Client’ are two different components of application. Admin side is used by system admins to manage users, contractors or skills. However, client side of application is used by contractors to view and modify their placements, view their current information and skills etc. This document would design and implement various reusable components to be utilized multiple times by admin or client side of application.

# Reusable Code

There are obvious benefits of reusing codes. It makes programming more cost effective. It helps to save time as less effort is required to achieve result that has already been implemented in another module. This helps to build more sophisticated application with much less effort. Another key benefit of code reuse is once one set of code is programmed and tested, when reused less testing is required as these codes are already tested and verified. PlaceU application uses various functions and procedure that can be reused and helps saves time.

## Function and Procedure

Functions and procedures offers ability to commonly used codes to be programmed and tested once and reused by different components of application as required. Functions and procedure are compiled and stored as database object that can be accessed via different components of application using object name. Both functions and procedures are capable of accepting arguments however, only functions can return value. However, both functions and procedure can return a value as output argument. Keeping codes that are required to be used quite often can be stored as function or procedure. These components can later be accessed when required. Example of use of reusable code is use of same function every time to retrieve contractor’s current salary.

# Implementation Method

## Package

A package in oracle database is utilized to group related PL objects such as variables, function and procedures. There are two parts of package, first is called specification and other is called body. Specification of package declares types, variables etc. while body of package defines them. Components inside a package can be access using object name together with ‘.’(dot) and name of package. Body of package can be modified without making affect to specification if input and output is not changed.

Use of security functionalities and reusable codes are implemented in PlaceU application via package. Reusable components such functions or procedures are first decaled in package specification. Later in the body of package, these functions and procedures are defined. This helps to manage components and keep codes organized and cleaned. These components are later consumed by other components from admin or client side of application.

# Implementation

## Package Specification

create or replace package "LDS\_PKG" is   
PROCEDURE SkilAudit(Action in varchar2,OldData varchar2,NewData varchar2);  
FUNCTION get\_sal(Id IN NUMBER) RETURN NUMBER;  
FUNCTION getMaxJoinDate(Id IN NUMBER) RETURN DATE;  
FUNCTION getCurrentCompany(Id IN NUMBER) RETURN VARCHAR2;  
FUNCTION getSkillNmae(SkillId IN NUMBER) RETURN VARCHAR2;  
PROCEDURE proc\_my\_placement (Id NUMBER, result\_refcur IN OUT SYS\_REFCURSOR);  
END LDS\_PKG;

Codes above builds package specification for PlaceU database application. It consists of various procedures and functions. These components are consumed from within or outside the package. Body for the packages is provided below.

## Package Body

Package body for the give LDS\_PKG package defines the components declared in package specification. The core purpose is to demonstrate use of reusable / sharable code. This document describes development of reusable of codes and their use in multiple scenarios.

### Function to GETMAXJOINDATE (Id IN NUMBER)

Getting latest placement join date for a contractor is required quite often. Hence putting the code inside function would help to reuse them with ease.

FUNCTION getMaxJoinDate(Id IN NUMBER)  
return date is  
  lds\_date date;  
begin  
select (MAX(PTL\_ACTUAL\_START\_DATE))  
    into lds\_date  
    from LDS\_PLACEMENT where FK2\_CONTRACTOR\_ID=Id;  
  return lds\_date;  
end;

#### Multiple use of getMaxJoinDate

1. In function getCurrentCompany

First use of getMaxJoinDate is done while getting currect company details. This is via use of contractor id and getMaxJoinDate to identify latest placement row then get company name.

FUNCTION getCurrentCompany(Id IN NUMBER)  
return varchar2 is  
company varchar2(50);  
begin  
SELECT account.ACC\_NAME into company  
FROM LDS\_ACCOUNT account  
INNER JOIN LDS\_PLACEMENT placement  
ON account.ACCOUNT\_ID = placement.FK1\_ACCOUNT\_ID  
where placement.FK2\_CONTRACTOR\_ID =Id  
AND placement.PTL\_ACTUAL\_START\_DATE=getMaxJoinDate(Id);  
return company;  
end;

1. In My Current info report

In My current info page for client side application, a report is generated to display latest join date, salary and company name of logged contractor. GetMaxJoinDate is utilized here to get joining date of latest placement of the contractor.

select

LDS\_PKG.getMaxJoinDate(CONTRACTOR\_ID) JoinDate,

LDS\_PKG.get\_sal(CONTRACTOR\_ID) salary,

LDS\_PKG. getCurrentCompany(CONTRACTOR\_ID) Company

from "#OWNER#"."LDS\_CONTRACTOR"

WHERE CONTRACTOR\_ID in (select CONTRACTOR\_ID FROM PLACEUSER where upper(USERNAME)=upper(:APP\_USER))

### Function GETSKILLNMAE (SKILLID IN NUMBER)

There are multiple occasions when skill description is required to be displayed on application page rather than their Id. It would be helpful to build a reusable component (function) that can return description of skill based on their Id.

FUNCTION getSkillNmae(SkillId IN NUMBER)  
return varchar2 is  
skillname varchar2(50);  
begin  
SELECT SKILL\_DESC into skillname  
FROM LDS\_SKILL where SKILL\_ID    =SkillId;  
return skillname ;  
end;

#### Multiple use of getSkillNmae

1. In All contractor report (Admin Panel)

Admin Panel of the application consists of an All contractors page which lists all contractors with in the application. getSkillNmae function has been utilized to display skill name rather than the skill Id.

select "CONTRACTOR\_ID",

"CON\_NAME",

"CON\_POSTCODE",

LDS\_PKG.getSkillNmae(CON\_SKILL\_1) Skill1,

LDS\_PKG.getSkillNmae(CON\_SKILL\_2) Skill2,

LDS\_PKG.getSkillNmae(CON\_SKILL\_3) Skill3,

"HIGHEST\_QUAL",

"PREFERRED\_ROLE",

"CREATEDBY",

"CREATEDDATE"

from "#OWNER#"."LDS\_CONTRACTOR"

1. In My Skills report (Contractor Panel)

After logging into the client side of application, contractor can navigate to My Skill page which would display their skills. To display skill name instead of skill id getSkillNmae function has been reused.

select

LDS\_PKG.getSkillNmae (CON\_SKILL\_1) skill1,

LDS\_PKG.getSkillNmae (CON\_SKILL\_2) skill2,

LDS\_PKG.getSkillNmae (CON\_SKILL\_3) skill3

from "#OWNER#"."LDS\_CONTRACTOR"

where CONTRACTOR\_ID in (select CONTRACTOR\_ID FROM PLACEUSER where upper(USERNAME)=upper(:APP\_USER))

### Procedure PROC\_MY\_PLACEMENT

Procedure is another way of saving compiled code as database object. PROC\_MY\_PLACEMENT procedure would take contractor Id as Input and provide curor with placement list for that Id as output. Instead of running whole code each time placement detail is required, calling procedure is much cleaner and manageable.

PROCEDURE proc\_my\_placement (Id NUMBER, result\_refcur IN OUT SYS\_REFCURSOR)  
IS  
BEGIN  
OPEN result\_refcur FOR SELECT \* FROM LDS\_PLACEMENT WHERE FK2\_CONTRACTOR\_ID= Id;  
END;

#### Multiple use of PROC\_MY\_PLACEMENT

1. In report to display past placements duration history

PROC\_MY\_PLACEMENT procedure is first used in a PL/SQL block that generates report to display past placements of given contractor and duration of each placement. As shown in code below, PROC\_MY\_PLACEMENT accepts contractor Id as input and outputs ref\_cursor consisting of placement details. This placement is looped to generate required report.

DECLARE

Id NUMBER :=10;

PlaceId NUMBER;

act\_end\_date DATE;

act\_start\_date DATE;

row\_date LDS\_PLACEMENT%rowtype;

result\_refcur SYS\_REFCURSOR;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Contractor ID ' || Id);

DBMS\_OUTPUT.PUT\_LINE('PlaceId Start Date End Date Total Length');

DBMS\_OUTPUT.PUT\_LINE('------- ---------- -------- ------------');

LDS\_PKG.proc\_my\_placement(Id, result\_refcur);

LOOP

FETCH result\_refcur INTO row\_date;

EXIT WHEN result\_refcur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(row\_date.PLACEMENT\_ID

|| ' ' || row\_date.PTL\_ACTUAL\_START\_DATE

|| ' ' || row\_date.PLT\_ACTUAL\_END\_DATE

|| ' ' || (to\_date(row\_date.PLT\_ACTUAL\_END\_DATE,'dd/mm/yyyy')- to\_date(row\_date.PTL\_ACTUAL\_START\_DATE,'dd/mm/yyyy'))||' Days'

);

END LOOP;

CLOSE result\_refcur;

END;

1. In report to display past placement salary history

PROC\_MY\_PLACEMENT procedure is later reused in a PL/SQL block that generates report to display past placements of given contractor and salary comparison with max/min salary for each placement. As shown in code below, PROC\_MY\_PLACEMENT accepts contractor Id as input and outputs ref\_cursor consisting of placement details. This placement is looped to generate required report.

DECLARE

Id NUMBER :=10;

PlaceId NUMBER;

act\_end\_date DATE;

act\_start\_date DATE;

row\_data LDS\_PLACEMENT%rowtype;

result\_refcur SYS\_REFCURSOR;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Contractor ID ' || Id);

DBMS\_OUTPUT.PUT\_LINE('PlaceId Max Salary Min Salary Less Than Max More than Min Actual Salary');

DBMS\_OUTPUT.PUT\_LINE('------- ---------- -------- ------------ ------------- -------------');

LDS\_PKG.proc\_my\_placement(Id, result\_refcur);

LOOP

FETCH result\_refcur INTO row\_data;

EXIT WHEN result\_refcur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(row\_data.PLACEMENT\_ID

|| ' ' || row\_data.MAX\_SALARY

|| ' ' || row\_data.MIN\_SALARY

|| ' ' || (row\_data.MAX\_SALARY-row\_data.ACTUAL\_SALARY)

|| ' ' || (row\_data.ACTUAL\_SALARY-row\_data.MIN\_SALARY)

|| ' ' || row\_data.ACTUAL\_SALARY

);

END LOOP;

CLOSE result\_refcur;

END;

### Procedure SKILAUDIT

Another procedure build to demonstrate reusable code is SKILAUDIT procedure. This procedure would take action name, new data and old data as in parameter to insert audit log in skill interaction audit table.

Procedure SkilAudit(Action in varchar2,OldData varchar2,NewData varchar2)IS  
BEGIN  
Insert into skillInteractionAudit(action,oldData,newData) values(Action,OldData,NewData);  
END;

#### Multiple use of SKILAUDIT

1. To audit update
2. To audit delete

Procedure SKILAUDIT is reused multiple times inside skillUpdateDelete\_audit trigger. Each time update or delete action is performed on skill table, this procedure would be utilized to keep audit log as shown in code below.

create or replace trigger skillUpdateDelete\_audit  
after update or delete  
on LDS\_SKILL  
for each row  
Declare  
v\_id number;  
begin  
If updating THEN  
Begin  
If (:Old.SKILL\_DESC<> :New.SKILL\_DESC) Then  
begin  
lds\_pkg.SkilAudit('Update',to\_char(:old.SKILL\_DESC),to\_char(:new.SKILL\_DESC));  
end;  
end if;  
END;  
End If;  
If deleting THEN  
begin  
lds\_pkg.SkilAudit('Delete',to\_char(:old.SKILL\_DESC),NULL);  
end;  
End if;  
end;

# Testing

Testing is one of integral stage of developing reusable components. It is essential to build components that does what it’s intended to do and are free of bugs. This helps to increase efficiency and productive of application. Tests of developed reusable components (functions and procedures) are tested via use of test plans and test logs.

# Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Case No.** | **Test Objective** | **Precondition** | **Steps** | **Test** | **Expected Result** |
| 1. | Use getMaxJoinDate function to get latest placement join date | Login contractor page | Client side module 🡪 My current info | Navigate to My current info page to view current join date | Report page should use getMaxJoinDate function in query successfully |
| 2. | Use getMaxJoinDate to get current company info | Login contractor page | Client side module 🡪 My current info | Navigate to My current info page to view current join date | Function getCurrentCompany should use getMaxJoinDate function to get current company of contractor |
| 3. | Use of getSkillNmae in Contractor module | Login contractor page | Client side module 🡪 My Skills | Navigate to My Skills page to view skills of logged in contractor | My Skills report page should use getSkillNmae function to list skills of contractor |
| 4. | Use of getSkillNmae in Admin module | Login Admin module | Admin side module 🡪 All contractors | Navigate to All contractor’ page to get report of contractor with skill description | All Contractors page should use getSkillNmae function to get name of skill in contractor’s report |
| 5. | Trigger should use SkilAudit procedure to audit skill table update | Login Admin module | Skill management 🡪 Edit | Update a row in skill table | Audit log should be inserted in skill audit table of type update |
| 6. | Trigger should use SkilAudit procedure to audit skill table delete | Login Admin module | Skill management 🡪 Delete | Delete a row in skill table | Audit log should be inserted in skill audit table of type delete |
| 7. | proc\_my\_placement should return cursor consisting all placement of given contractor Id | Call proc\_my\_placement with an PL/SQL stamen with correct arguements | proc\_my\_placement (input, output) | Call procedure from PL block to generate various report | PL/SQL block should use procedure multiple times to generate various reports |

# Test Log

#### Test Case 1

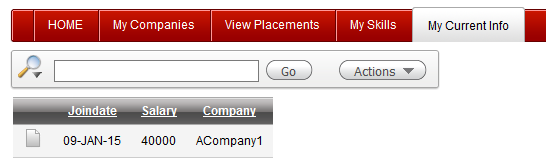


Figure Join Date of Latest placement using function getMaxJoinDate

#### Test Case 2

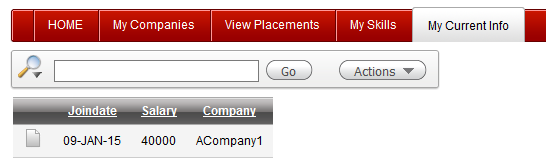


Figure Company Name of Latest palcement using function getMaxJoinDate in getCurrentCompany function

#### Test Case 3

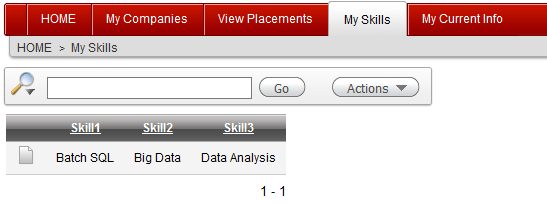


Figure Successfully listed skills of logged in contractor using function getSkillNmae

#### Test Case 4

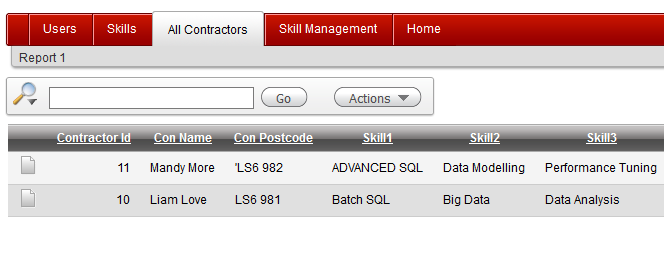


Figure All contractors page successfully generates report with skill names function getSkillNmae

#### Test Case 5



Figure Modifying skill description in Skill management page 🡪 apply changes

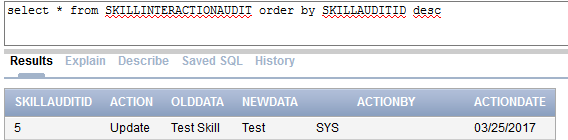


Figure skill table update log was successfully audited

#### Test Case 6



Figure Delete skill from skill management page 🡪 Delete

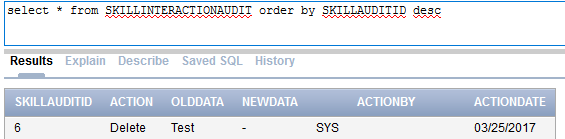


Figure skill table delete log was successfully audited

#### Test Case 7

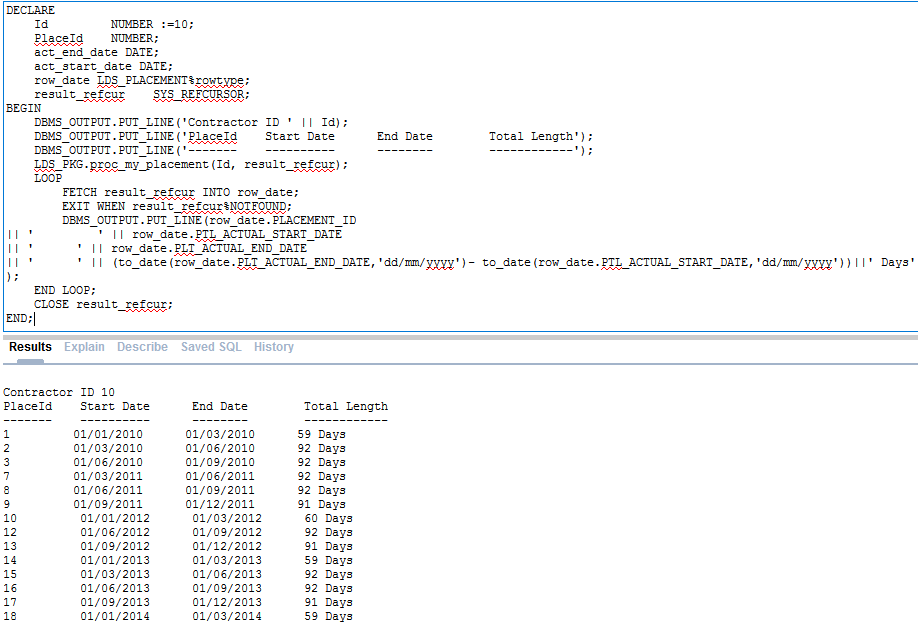


Figure Successfully used My\_Placement procedure to get report on previous placments duration in days

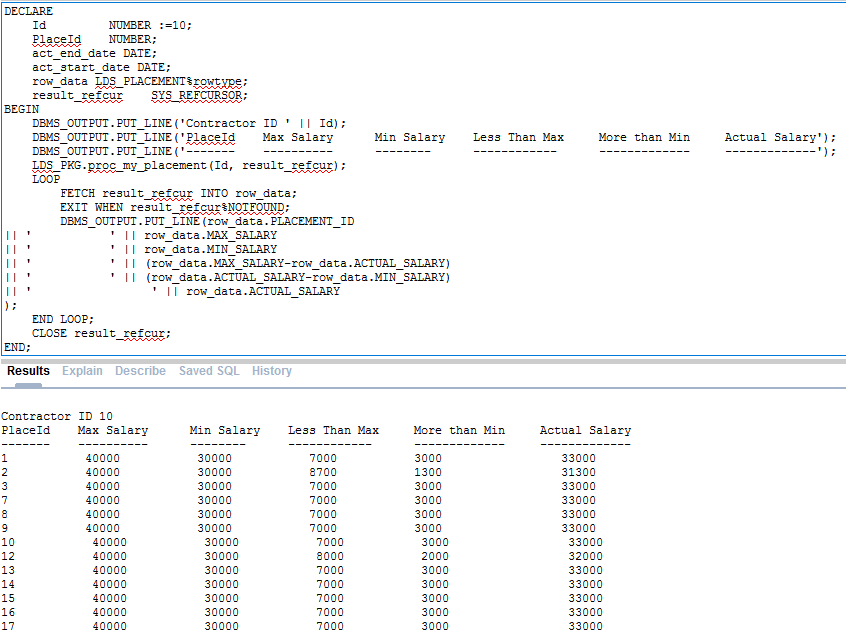


Figure Succesfully reused procedure my\_placements to generate report comparing actual salary with min/max salary

# Test Log Table

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Expected Result | Actual Result | Date |
| 1 | Report page should use getMaxJoinDate function in query successfully | Successfully used getMaxJoinDate to query latest placement join date | March 22 2017 |
| 2 | Function getCurrentCompany should use getMaxJoinDate function to get current company of contractor | Successfully used getMaxJoinDate to query current company name | March 22 2017 |
| 3. | My Skills report page should use getSkillNmae function to list skills of contractor | getSkillNmae was consumed successfully to get list of skills | March 23 2017 |
| 4. | All Contractors page should use getSkillNmae function to get name of skill in contractor’s report | getSkillNmae component was used successfully to display skill name in report | March 23 2017 |
| 5. | Audit log should be inserted in skill audit table of type update | Skill audit procedure was used successfully to audit skill table update action | March 23 2017 |
| 6. | Audit log should be inserted in skill audit table of type delete | Skill audit procedure was used successfully to audit skill table delete action | March 23 2017 |
| 7. | proc\_my\_placement should output cursor consisting all placement of given contractor Id | proc\_my\_placement successfully gave output cursor that was utilized multiple times to generate various reports | March 23 2017 |

# Course Work 3

# Introduction

Database performance tuning is simply improvement of overall performance of database query capability. Most of the databases show decrement in performance as the size of the database grows. It is fundamental requirement for any database application to plan performance tuning from the beginning and take necessary actions as the need of performance tuning grows.

Database tuning is related to not only query optimization but also related to setting up database environment and design of database system.

# Growing data for demonstration

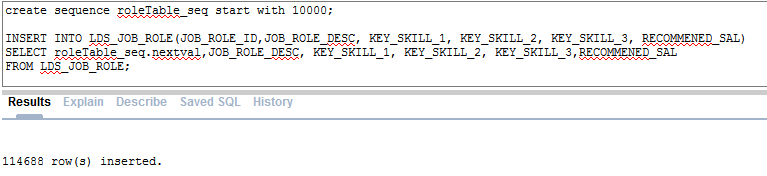
For the demonstration purpose, more than 100,000 data are inserted into LDS\_JOB\_ROLE table. Code for injecting data into table is provided below.

create sequence roleTable\_seq start with 10000;

INSERT INTO LDS\_JOB\_ROLE(JOB\_ROLE\_ID,JOB\_ROLE\_DESC, KEY\_SKILL\_1, KEY\_SKILL\_2, KEY\_SKILL\_3, RECOMMENED\_SAL)

SELECT roleTable\_seq.nextval,JOB\_ROLE\_DESC, KEY\_SKILL\_1, KEY\_SKILL\_2, KEY\_SKILL\_3,RECOMMENED\_SAL

FROM LDS\_JOB\_ROLE;



# Performance Tuning Strategies

1. Distributed installation Locations

Plan of proper location plays important role in database performance. It is considered bad practice to set oracle base, database file location and software location in same drive. Distribution of these components into different location can help to reduce I/O performance hence improving the database performance specially when there is huge number of data is stored and large number of transactions requires to be done. It is important to distribute control files, data files, temp files and other files such as log files, server parameter files etc.

1. Database Design

Another fundamental stage for performance tuning of database is database design itself. It is key aspect of any database application development to design database properly with proper keys, index, tables and constraints. PlaceU database application has created tables with primary key in each table means all tables has primary index. Schema is basically database object that helps to improve data retrieval via use of pointer. Having primary key in each table mean each table automatically does have index.

1. SQL query Performance

Finally query optimization is final important aspect of database performance tuning. While joining tables together with where clause, it is important to make sure join is done first before where clause as it improve the quality of that query. It is important to analyze a query statement then make changes accordingly to optimized the performance of that query. This document utilizes Oracle explanation plan to analyze query and understand their execution structure. This would help developer to plan and make necessary changes on table design, index or query itself.

# Execution Plan

#### Explain 1

Following code is from function Get\_Sal() to get contractor’s latest placement salary. Hash join is formed between placement table. Execution plan for the code has been presented below.

SELECT to\_number(placement.ACTUAL\_SALARY) FROM

contractorproject.LDS\_PLACEMENT placement

JOIN

( SELECT FK2\_CONTRACTOR\_ID,MAX(PTL\_ACTUAL\_START\_DATE) AS col\_date

FROM contractorproject.LDS\_PLACEMENT

WHERE FK2\_CONTRACTOR\_ID=10

GROUP BY FK2\_CONTRACTOR\_ID

) m

ON m.FK2\_CONTRACTOR\_ID = placement.FK2\_CONTRACTOR\_ID

AND m.col\_date = placement.PTL\_ACTUAL\_START\_DATE;

#### Without index

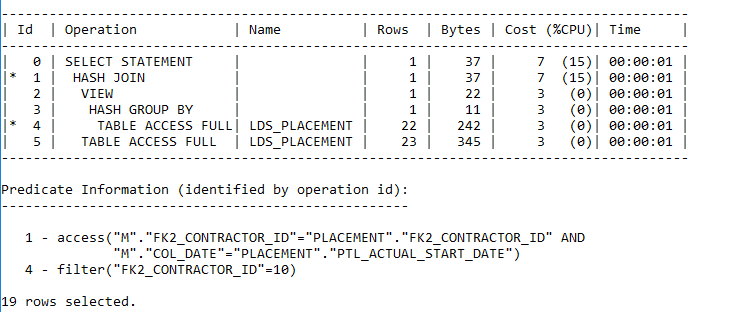


Figure Explanation plan without indexing

Figure above shows execution plan for given query where table is not indexed properly. Table access is full on LDS\_PLACEMENT due to lack of indexing.

#### With Index

As where clause is being used on actual date column of placement table, new index\_actualdate index is created using command below.

Create index INDEX\_ACTUALDATE ON lds\_placement(PTL\_ACTUAL\_START\_DATE);

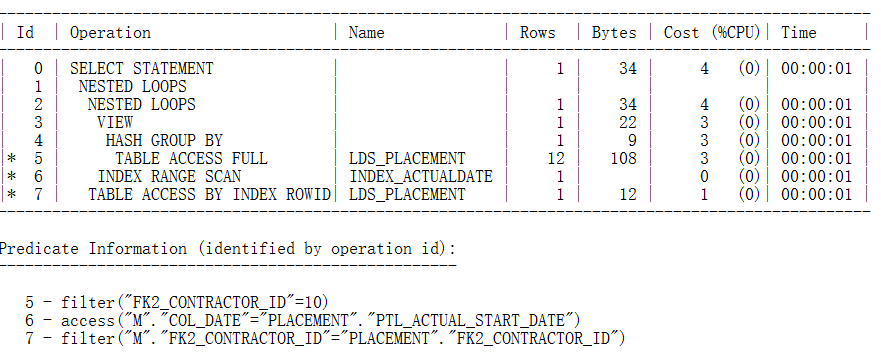


Figure Explanation plan with indexing

Figure above shows new execution plan for given query. Figure shows improved CPU cost. Query now using INDEX\_ACTUALDATE index to find the information. Finally table is accessed by ROWID index which helps to reduce query cost.

#### Explain 2

SELECT SKILL\_DESC

FROM contractorproject.LDS\_SKILL where SKILL\_ID =3

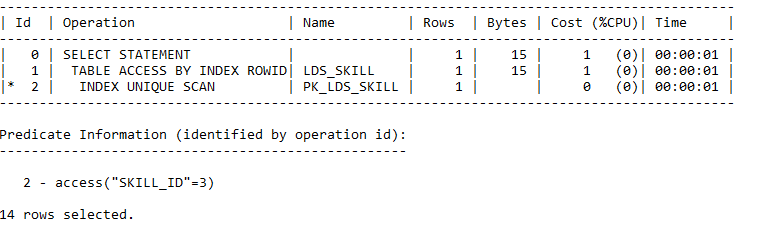


Figure Execution plan for query with unique index

Image above shows execution plan for query, here, table LDS\_SKILL has primary index PK\_LDS\_SKILL. This index was auto generated when primary key for table was created. Where clause on query above is on column with primary index, this means query is already optimized at some level. No changes are necessary in above code.

#### Explain 3

select (MAX(PTL\_ACTUAL\_START\_DATE)) StartDate

from contractorproject.LDS\_PLACEMENT where FK2\_CONTRACTOR\_ID=10;

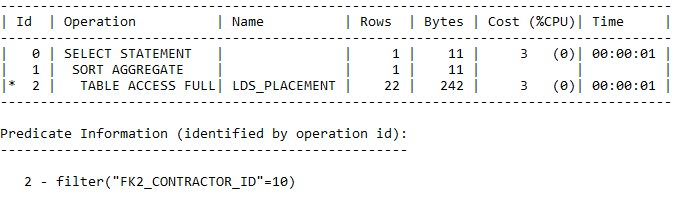


Figure Execution plan without index

Query above is taken from function to get join date of latest placement for a contractor. Figure shows full table access during execution of where clause. This is due to absence of any index on that particular column.

#### With index

create index index\_placement\_contractor on lds\_placement(FK2\_CONTRACTOR\_ID);

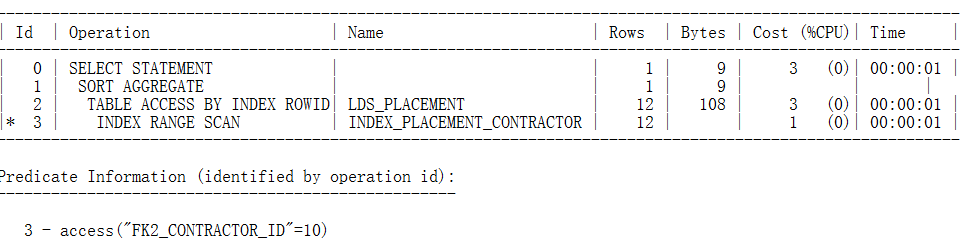


Figure Execution plan with index

An index on FK2\_CONTRACTOR\_ID column has been created hence new execution plan show where clause uses range scan instead of full table scan. This improve the Bytes of query hence improving the performance.

# Recommendation

While it is extremely important to maintain performance of database by database tuning, it is always useful to keep back up of database in case of data disaster. This can be achieved via export and import of dump file. Code below demonstrates exporting and importing of database back up file of contractor project schema.

## Export/Import Dump File

CREATE OR REPLACE DIRECTORY test\_dir AS 'e:\Database';

GRANT READ, WRITE ON DIRECTORY test\_dir TO contractorproject;

Expdp contractorproject/contractorproject@db10g schemas=contractorproject directory=TEST\_DIR dumpfile=contractorproject.dmp logfile=expdpcontractorproject.log

Impdp contractorproject/contractorproject@db10g schemas=contractorproject directory=TEST\_DIR dumpfile=contractorproject.dmp logfile= impdpcontractorproject.log



Figure Exporting backup file for CONTRACTORPROJECT schema

# Summary

This document explored reusable code and performance tuning of database for PlaceU database application. Use of reusable components helps to improve quality of application by keeping codes clean and manageable. This also helps to make testing process easier. Each reusable component such as function or procedure are tested after they are built hence developer does not required to test these each time they are implemented. This makes development phase shorter.

PlaceU application has planned and built functions to return join date of latest placement of contractor and function to return skill name based on their skill id. Document has also planned and build procedures to keep audit log for skill table and return placements of a contractor. Document has demonstrated use of these components from different location.

Document has also access execution plan for PlaceU application while analyzing requirement of indexes to improve query performance. Various indexes were added to improve query performance. However, it is important to keep in mind that excessive use of index in unnecessary columns can lead to decrement in performance as index reduces the insert performance while improving query performance.