# Database Application Development

# Project/Assignment

## Milestone 1 (part 1)

## Objective:

In this assignment, you create a simple HR application using the C++ programming language and Oracle server. This assignment helps students learn a basic understanding of application development using C++ programming and an Oracle database

## Submission:

***This Milestone is a new project that simply uses what was learned in the SETUP.***

***Your submission will be a single text-based .cpp file including your C++ program for the Database Application project/assignment. The file must include a comment header of student name and ID number.***

```
AS_Part1.txt
```

Your submission needs to be FULLY commented.

Note: you can submit combine code of part1 and part2 as final.txt also

## Instruction:

In this assignment, we use the same database that you use for your labs.

### *Connecting to an Oracle database from a C++ Program*

In your function ***main()***, create a connection to your database.

You need to implement the following functions:

***int menu(void);***

The ***menu()*** function returns an integer value which is the selected option by the user from the menu. This function displays the following menu options:

1) Find Employee
2) Employees Report
3) Add Employee

4) Update Employee
5) Remove Employee
6) Exit

Before printing the menu, display the following title on the screen

********** HR Menu **********

Prompt the user to enter an option. If the user enters an incorrect option, the user is asked to enter an option again. When the user enters a correct option (1 to 5), the function returns the selected value.

If the user selects 6 (Exit), the program terminates.

## *int findEmployee(*conn, int employeeNumber, struct Employee *emp);*

This function receives a connection object, an integer number as the employee number, and a pointer to a variable of type Employee. The function returns 0 if the employee does not exist. It returns 1 if the employee exits.
To store the employee data from the ***findEmployee()*** function, we use a variable of type structure called Employee. The Employee structure has the following members:

```
struct Employee{
      int employeeNumber;
      char lastName[50];
      char firstName[50];
      char email[100];
      char phone[50];
      char extension[10];
      char reportsTo[100];
      char jobTitle[50];
      char city[50];

};
```

The *ReportsTo* member stores the first name and the last name of the employee who is the manager of the given employee number.
If the employee exists, store the employee data into the members of an Employee variable using the third parameter in the ***findEmployee()*** function which references to that variable of type Employee.

**Note**: For this report, you may need to query more than one table (join).

## *void displayEmployee(*conn, struct Employee emp);*

If the user selects option 1, this function is called. First, prompt the user to enter a value for the employee number. Then, call function ***findEmployee()*** to check if the employee with the given employee number exists. If the returning value of function ***findEmployee()*** is 0, display a proper error message.
Sample error message:

Employee 1122 does not exist.

Otherwise, call the function ***displayEmployee()*** to display the employee information.
This function receives a connection pointer and values of a variable of type Employee and displays all members of the emp parameter.

Display the employee information as follows:

employeeNumber = 1002
lastName = Murphy
firstName = Diane
email = dmurphy@classicmodelcars.com
phone = +1 650 219 4782
extension = x5800
reportsTo =
jobTitle = President
city = San Francisco

## *void displayAllEmployees(*conn);*

If the user selects option 2 (Employees Report), call function ***displayAllEmployees().***
This function receives a connection pointer and displays all employees' information if exists.

Write a query to select and display the following attributes for all employees.

| E | Employee Name | Email | Phone | Ext | Manager |
|---|---|---|---|---|---|
| 1002 | Diane Murphy | dmurphy@classicmodelcars.com | +1 650 219 4782 | x5800 | |
| 1056 | Mary Patterson | mpatterso@classicmodelcars.com | +1 650 219 4782 | x4611 | Diane Murphy |
| 1076 | Jeff Firrelli | jfirrelli@classicmodelcars.com | +1 650 219 4782 | x9273 | Diane Murphy |
| 1143 | Anthony Bow | abow@classicmodelcars.com | +1 650 219 4782 | x5428 | Mary Patterson |
| 1165 | Leslie Jennings | ljennings@classicmodelcars.com | +1 650 219 4782 | x3291 | Anthony Bow |
| 1166 | Leslie Thompson | lthompson@classicmodelcars.com | +1 650 219 4782 | x4065 | Anthony Bow |
| 1188 | Julie Firrelli | jfirrelli@classicmodelcars.com | +1 215 837 0825 | x2173 | Anthony Bow |
| 1216 | Steve Patterson | spatterson@classicmodelcars.com | +1 215 837 0825 | x4334 | Anthony Bow |
| 1286 | Foon Yue Tseng | ftseng@classicmodelcars.com | +1 212 555 3000 | x2248 | Anthony Bow |
| 1323 | George Vanauf | gvanauf@classicmodelcars.com | +1 212 555 3000 | x4102 | Anthony Bow |
| 1102 | Gerard Bondur | gbondur@classicmodelcars.com | +33 14 723 4404 | x5408 | Mary Patterson |
| 1337 | Loui Bondur | lbondur@classicmodelcars.com | +33 14 723 4404 | x6493 | Gerard Bondur |
| 1370 | Gerard Hernandez | ghernande@classicmodelcars.com | +33 14 723 4404 | x2028 | Gerard Bondur |
| 1401 | Pamela Castillo | pcastillo@classicmodelcars.com | +33 14 723 4404 | x2759 | Gerard Bondur |
| 1702 | Martin Gerard | mgerard@classicmodelcars.com | +33 14 723 4404 | x2312 | Gerard Bondur |
| 1621 | Mami Nishi | mnishi@classicmodelcars.com | +81 33 224 5000 | x101 | Mary Patterson |
| 1625 | Yoshimi Kato | ykato@classicmodelcars.com | +81 33 224 5000 | x102 | Mami Nishi |
| 1088 | William Patterson | wpatterson@classicmodelcars.com | +61 2 9264 2451 | x4871 | Mary Patterson |
| 1611 | Andy Fixter | afixter@classicmodelcars.com | +61 2 9264 2451 | x101 | William Patterson |
| 1612 | Peter Marsh | pmarsh@classicmodelcars.com | +61 2 9264 2451 | x102 | William Patterson |
| 1619 | Tom King | tking@classicmodelcars.com | +61 2 9264 2451 | x103 | William Patterson |
| 1501 | Larry Bott | lbott@classicmodelcars.com | +44 20 7877 2041 | x2311 | Gerard Bondur |
| 1504 | Barry Jones | bjones@classicmodelcars.com | +44 20 7877 2041 | x102 | Gerard Bondur |

**Note**: For this report, you may need to query more than one table (join).

If the query does not return any rows, display a proper message:

There is no employees' information to be displayed.

**Note**: For each query in your assignment, make sure you handle the errors and display the proper message including the error_code.

Error_code is a number returned if the query execution is not successful.

# FINAL SUBMISSION (Part 2)

## Instruction:

In this assignment, we complete the application from the first part1 (milestone1) to insert, update, and delete the employee information.

You need to implement the following functions:

### *void insertEmployee(*conn, struct Employee emp);*

This function receives a connection pointer and a structure of type Employee and inserts the given employee information stored in the parameter ***emp*** to employee table.
In function ***insertEmployee()***, call the function ***findEmployee()*** to see if the employee number of the given employee exists. If an employee with the same employee number exists display a proper message:
"An employee with the same employee number exists." and return to the menu.

Otherwise, insert the employee information into the employee table and display the following message:

"The new employee is added successfully."

**Note**: For simplicity, assume that the office code of the new employees is 1 and the manager id (reportsTo) is 102 by default.

### *void updateEmployee(*conn, int employeeNumber);*

This function receives a connection pointer and an integer number as the employee number and updates the phone extension for the given employee. In function ***updateEmployee()***, call function ***findEmployee()*** to see if the employee exists in table employees. If employee does exist, ask the user to enter the new phone extension. Store the new extension in table employees for the given employee number.

### *void deleteEmployee(*conn, int employeeNumber);*

This function receives a connection pointer and an integer number as the employee number and deletes a row with the given employee number from table employees. In function ***deleteEmployee()***, call function ***findEmployee()*** to see if the employee with the given employee number exists. If the employee does not exist display a proper message:
"The employee does not exist."
If the employee exits, delete the row from table employees and display a proper message:
"The employee is deleted."

## Function *main()*

From the menu in the first part of the assignment, complete options 3 to 5.

1) Find Employee
2) Employees Report
3) Add Employee
4) Update Employee
5) Remove Employee
**6)** Exit

## Add an Employee

If the user chooses option 3, prompt the user to enter the new employee information and store them into a variable of type Employee structure. Then, call function ***insertEmployee()*** to insert the new employee information in table employees.

```
Employee Number: 1818
Last Name: Adam
First Name: Sarah
Email: sadam@email.com
extension: x4411
Job Title: Sales Rep
City: Toronto
```

**NOTE:** You do not need to ask the user to enter values for the members *reportsTo* and *phone*.

## Update an Employee

If the user chooses option 4, ask the user to enter the employee number:

Employee Number: 1216

Then, call function ***updateEmployee()*** to update the phone extension for the row with the employee number **1216**. In this function, the user is asked to enter the new extension:

New Extension: x2111

The extension column of the row with the employee number 1216 will be updated with the new value x2111.

## Delete an Employee

If the user chooses option 5, ask the user to enter the employee number:

Employee Number: 1818

Then, call function ***deleteEmployee()*** to remove the employee from table employees.

**Note**: For each query in your assignment, make sure you handle the errors and display the proper message including the error_code.

Error_code is a number returned if the query execution is not successful.

**Offices Table**

| | OFFICECODE | CITY | PHONE | ADDRESSLINE1 | ADDRESSLINE2 | STATE | COUNTRY | POSTALCODE | TERRITORY |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | San Francisco | +1 650 219 4782 | 100 Market Street | Suite 300 | CA | USA | 94080 | NA |
| 2 | 2 | Boston | +1 215 837 0825 | 1550 Court Place | Suite 102 | MA | USA | 02107 | NA |
| 3 | 3 | NYC | +1 212 555 3000 | 523 East 53rd Street | apt. 5A | NY | USA | 10022 | NA |
| 4 | 4 | Paris | +33 14 723 4404 | 43 Rue Jouffroy 'abbans | (null) | (null) | France | 75017 | EMEA |
| 5 | 5 | Tokyo | +81 33 224 5000 | 4-1 Kioicho | (null) | Chiyoda-Ku | Japan | 102-8578 | Japan |
| 6 | 6 | Sydney | +61 2 9264 2451 | 5-11 Wentworth Avenue | Floor #2 | (null) | Australia | NSW 2010 | APAC |
| 7 | 7 | London | +44 20 7877 2041 | 25 Old Broad Street | Level 7 | (null) | UK | EC2N 1HN | EMEA |

**Employees Table**

| | EMPLOYEENUMBER | LASTNAME | FIRSTNAME | EXTENSION | EMAIL | OFFICECODE | REPORTSTO | JOBTITLE |
|---|---|---|---|---|---|---|---|---|
| 1 | 1002 | Murphy | Diane | x5800 | dmurphy@classicmodelcars.com | 1 | (null) | President |
| 2 | 1056 | Patterson | Mary | x4611 | mpatterso@classicmodelcars.com | 1 | 1002 | VP Sales |
| 3 | 1076 | Firrelli | Jeff | x9273 | jfirrelli@classicmodelcars.com | 1 | 1002 | VP Marketing |
| 4 | 1088 | Patterson | William | x4871 | wpatterson@classicmodelcars.com | 6 | 1056 | Sales Manager (APAC |
| 5 | 1102 | Bondur | Gerard | x5408 | gbondur@classicmodelcars.com | 4 | 1056 | Sale Manager (EMEA) |
| 6 | 1143 | Bow | Anthony | x5428 | abow@classicmodelcars.com | 1 | 1056 | Sales Manager (NA) |
| 7 | 1165 | Jennings | Leslie | x3291 | ljennings@classicmodelcars.com | 1 | 1143 | Sales Rep |
| 8 | 1166 | Thompson | Leslie | x4065 | lthompson@classicmodelcars.com | 1 | 1143 | Sales Rep |
| 9 | 1188 | Firrelli | Julie | x2173 | jfirrelli@classicmodelcars.com | 2 | 1143 | Sales Rep |
| 10 | 1216 | Patterson | Steve | x4334 | spatterson@classicmodelcars.com | 2 | 1143 | Sales Rep |
| 11 | 1286 | Tseng | Foon Yue | x228 | ftseng@classicmodelcars.com | 3 | 1143 | Sales Rep |
| 12 | 1323 | Vanauf | George | x4102 | gvanauf@classicmodelcars.com | 3 | 1143 | Sales Rep |
| 13 | 1337 | Bondur | Loui | x6493 | lbondur@classicmodelcars.com | 4 | 1102 | Sales Rep |
| 14 | 1370 | Hernandez | Gerard | x2028 | ghernande@classicmodelcars.com | 4 | 1102 | Sales Rep |
| 15 | 1401 | Castillo | Pamela | x2759 | pcastillo@classicmodelcars.com | 4 | 1102 | Sales Rep |
| 16 | 1501 | Bott | Larry | x2311 | lbott@classicmodelcars.com | 7 | 1102 | Sales Rep |
| 17 | 1504 | Jones | Barry | x102 | bjones@classicmodelcars.com | 7 | 1102 | Sales Rep |
| 18 | 1611 | Fixter | Andy | x101 | afixter@classicmodelcars.com | 6 | 1088 | Sales Rep |
| 19 | 1612 | Marsh | Peter | x102 | pmarsh@classicmodelcars.com | 6 | 1088 | Sales Rep |
| 20 | 1619 | King | Tom | x103 | tking@classicmodelcars.com | 6 | 1088 | Sales Rep |
| 21 | 1621 | Nishi | Mami | x101 | mnishi@classicmodelcars.com | 5 | 1056 | Sales Rep |
| 22 | 1625 | Kato | Yoshimi | x102 | ykato@classicmodelcars.com | 5 | 1621 | Sales Rep |
| 23 | 1702 | Gerard | Martin | x2312 | mgerard@classicmodelcars.com | 4 | 1102 | Sales Rep |