**DBS311NFF Group 1 Project - College Registration**

**Mohammadhossein Sobhanmanesh 116523200 -> Financial Table**

**Muhammad Ahmed 146908207 -> Section Table**

**Chungon Tse 154928188 -> Course Table**

# **Overview**

This application is on college registration and the focus is on the tables Financial, Section, and Course. It displays a menu of 13 options, which are 4 functions for each table and an exit option. The user input starts with picking an option (0 to 12) and then performing create, read, update, delete, and display all functions depending on each table’s constraints and expected input. Selecting 0 in the menu means quitting the application.

**User Guide – Table of Contents**

[Financial Table 2](#_Toc100219840)

[Create 2](#_Toc100219841)

[Read 2](#_Toc100219842)

[Update 2](#_Toc100219843)

[Delete 2](#_Toc100219844)

[Display all financial accounts 2](#_Toc100219845)

[Section Table 3](#_Toc100219846)

[Create 3](#_Toc100219847)

[Read 4](#_Toc100219848)

[Update 5](#_Toc100219849)

[Delete 6](#_Toc100219850)

[Display all course sections 6](#_Toc100219851)

[Course Table 7](#_Toc100219852)

[Create 7](#_Toc100219853)

[Read 8](#_Toc100219854)

[Update 9](#_Toc100219855)

[Delete 10](#_Toc100219856)

[Display all courses 10](#_Toc100219857)

[Presentation 10](#_Toc100219858)

# **Financial Table**

Financial table contains

* financialID (number and primary key) – bank account number
* balance (double) – the amount of money in a bank account
* bankName (varchar2) – name of the bank where an account is at
* bankAccNum (number) – the bank account number
* cardNum (number) – the card number for the account

financialID is a foreign key in the professor and student tables.

## Create

Insert a row in the financial table for all five columns. balance and bankName are mandatory.

## Read

Correct input of financialID will help retrieve a row. The procedure produces an error code which is 0 if input fails or greater than 0 if input succeeds.

## Update

Correct input will help update any of the columns. The procedure produces an error code which is 0 if input fails or greater than 0 if input succeeds.

## Delete

Correct input of financialID will help delete a row. The procedure produces an error code which is 0 if input fails or greater than 0 if input succeeds.

## Display all financial accounts

This is a select-all statement and displays all the rows in this table.

# **Section Table**

Section table contains

* sectionID (number and primary key) – course section ID
* professorID (number) – professor ID which is a foreign key from the professor table
* courseID (number) – course ID which is a foreign key from the course table
* classNumber (number) – class number for a certain course
* courseTime (varchar2) – when a course takes place every week

professorID and courseID are foreign keys in professor and course tables respectively.

## Create

This procedure asks for 5 input arguments and inserts a row in the section table. The values sectionID, classNumber and courseTime are mandatory and the values professorID and courseID must be values from their respective tables or the program will send back an error.

Correct input will help insert a row with at most 5 columns. The procedure produces an error code which is 0 if no new row is added or greater than 0 if input succeeds.

Example non-saved procedural call:

DECLARE

err\_code INTEGER;

BEGIN

spsectioninsert(err\_code, 50, 7250, 105, 5, 'Tuesday');

END;

Sample output:

Text

Description automatically generatedTable

Description automatically generated

Sample error:

Text

Description automatically generated

## Read

Correct input of sectionID will help retrieve a row with all 5 columns (if applicable).

The procedure produces an error code which is 0 if no data can be exists for the provided section ID or greater than 0 if input succeeds. The error code may also be -1 if the procedure fails in some manner, such as multiple rows returned for the same sectionID.

Example non-saved procedural call:

DECLARE

err\_code INTEGER;

m\_sectionID INTEGER := 75;

m\_professorID INTEGER;

m\_courseID INTEGER;

m\_classNumber INTEGER;

m\_courseTime VARCHAR2(255 BYTE);

BEGIN

spsectionread(err\_code, m\_sectionID, m\_professorID, m\_courseID, m\_classNumber, m\_courseTime);

IF err\_code > 0

THEN DBMS\_OUTPUT.PUT\_LINE('SECTION: ' || m\_sectionID || ', ' || m\_professorID || ', ' || m\_courseID || ', ' || m\_classNumber || ', ' || m\_courseTime);

END IF;

END;

Sample output:

Text

Description automatically generated

Sample error:

Text

Description automatically generated

## Update

This procedure asks for 5 input arguments. The values sectionID, classNumber and courseTime are mandatory and the values professorID and courseID must be values from their respective tables or the program will send back an error. Correct input will help update all of the of the columns professorID, courseID, classNumber, and courseTime.

The procedure produces an error code which is 0 if no rows are updated or greater than 0 if input succeeds.

Example non-saved procedural call:

DECLARE

err\_code INTEGER;

BEGIN

spsectionupdate(err\_code, 50, 7259, 168, 9, 'Friday');

END;

Sample output:

Text

Description automatically generated Table

Description automatically generated

Sample error:

Text

Description automatically generated

## Delete

Correct input of sectionID will help delete a row. The procedure produces an error code which is 0 if no data can be exists for the provided section ID 0 if input succeeds. The error code may also be -1 if the procedure fails in some other technical manner.

Example non-saved procedural call:

DECLARE

err\_code INTEGER;

BEGIN

spsectiondelete(err\_code, 50);

END;

Sample output:

Text

Description automatically generated Table

Description automatically generated

Sample error:

Text

Description automatically generated

## Display all course sections

This is a select-all statement and displays all the rows in this table using DBMS\_OUTPUT, as requested by assignment instructions.

Example non-saved procedural call:

BEGIN

spsectiondisplayall();

END;

# **Course Table**

Course table contains

* courseid (number) (primary key) – course ID
* coursename (varchar2) – name of the course
* crsdescription (varchar2) – course description
* creditnumber (number) – number of credits of each course
* programid (number) – study program ID, which is also in the program table

programid is linked to the program table as a foreign key. In the create, read, update, and delete functions below, if the input is correct as to data types and constraints below, the procedure returns an “error code” of 1 passing into the .cpp file’s corresponding functions. Otherwise, with wrong input, the “error code” is -1 passing into those functions.

## Create

Insert a row in the course table for 5 columns. courseName and creditNumber are mandatory. Notice that programid entries are fixed in the given program table, so programid can only be 6060-6069. Correct input will help insert a row of at most 5 columns. With valid input, the “error code” returns 1. Otherwise, it is -1.

Non-saved procedural code:

DECLARE

result integer := 0;

BEGIN

spCourseInsertByID(188,'Introduction to Monty Python','To taste the best elderberries',4,6068,result);

dbms\_output.put\_line('Insertion successful.');

EXCEPTION

WHEN OTHERS

THEN

DBMS\_OUTPUT.PUT\_LINE ('An error occurred');

END;

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

## Read

Correct input of courseid will help retrieve a row of at most 5 columns. Each courseid is unique so it returns exactly one row with valid input, giving an error code of 1, and -1 if no row is returned.

Non-saved procedural code:

DECLARE

cseId number := 188;

cseName varchar2(255 byte);

cseDesc varchar2(255 byte);

cseCredit number;

pId number;

result integer := 0;

BEGIN

spCourseReadByID(cseId,cseName,cseDesc,cseCredit,pId,result);

DBMS\_OUTPUT.PUT\_LINE ('Course ID: ' || cseId || ', ' || cseName || ', ' || cseDesc || ', ' || cseCredit || ' credits, program ID: ' || pId);

EXCEPTION

WHEN OTHERS

THEN

DBMS\_OUTPUT.PUT\_LINE ('An error occurred');

END;

Graphical user interface, application

Description automatically generated

## Update

Correct input will help update any of the 5 columns coursename, crsdescription, creditnumber, and programid. This function does not update courseid, and while there can be a change of programid, programid values are restricted to what exists in the program table. With valid input, the “error code” returns 1. Otherwise, it is -1.

Non-saved procedural code:

DECLARE

cseId number := 188;

cseName varchar2(255 byte) := 'Sir Galahad of Camelot';

cseDesc varchar2(255 byte) := 'is looking for the Holy Grail';

cseCredit number := 5;

pId number := 6065;

result integer := 0;

BEGIN

spCourseUpdateByID(cseId, cseName, cseDesc, cseCredit, pId, result);

DBMS\_OUTPUT.PUT\_LINE ('Data for course ID ' || cseId || ' are now updated');

DBMS\_OUTPUT.PUT\_LINE ('Course ID: ' || cseId || ', ' || cseName || ', ' || cseDesc || ', ' || cseCredit || ' credits, program ID: ' || pId);

EXCEPTION

WHEN OTHERS

THEN

DBMS\_OUTPUT.PUT\_LINE ('An error occurred');

END;

Graphical user interface, application

Description automatically generated with medium confidence Graphical user interface, text, application

Description automatically generated

## Delete

Correct input of courseid will help delete a row. This succeeds only when the input courseid matches an existing courseid, and it returns “error code” of 1. The update function does not change courseid, so if the user inputs the wrong courseid and creates a row, this row should be deleted, and there should be a new entry through the create function.

Non-saved procedural code:

DECLARE

cseId number := 188;

result integer := 0;

BEGIN

spCourseDeleteByID(cseId, result);

DBMS\_OUTPUT.PUT\_LINE ('Data for course ID ' || cseId || ' are now deleted');

EXCEPTION

WHEN OTHERS

THEN

DBMS\_OUTPUT.PUT\_LINE ('An error occurred');

END;

Graphical user interface, text, application, email

Description automatically generated

## Display all courses

This is a select-all statement and displays all the rows in this table using DBMS\_OUTPUT, as requested by assignment instructions.

## **Presentation**

<https://youtu.be/eCo-DMN5Ue4>