

# School of Information Technology

## Department of Computer Science



### COS326 Database Systems: Practical 4 2022

**Release Date: 25 August 2022**

**Submission Date: 02 September 2022**

**Lecturer: Mr S.M Makura**

**Total: 50 Marks**

#### A. Objectives

1. Practice the use of advanced features of PostgreSQL ORDBMS.
2. Learn how to implement PL/pgSQL functions, triggers, and database constraints using triggers.

#### B. Submission Procedure:

You should have PostgreSQL installed on your computer in order to complete this practical. When you are done:

1. You must submit files, named:
  - a. **EERD.pdf** which contains the database design as an EER diagram. Your name and student number must appear in this document.
  - b. **CreateStatements.sql** which contains all statements necessary to create the database 'objects' i.e. user defined types, sequences, tables, functions and triggers.
  - c. **InsertQueries.sql** which contains all statements that add to the content of the database (INSERT statements).
  - d. **SelectQueries.sql** which contains all statements that provide reports from the database (SELECT statements)
  - e. **TestTriggerQueries.sql** which contains all statements for testing the triggers (INSERT, UPDATE, and DELETE statements).
  - f. Compress the above documents into an archive (zip file) and upload it to ClickUP using the Practical 3 submission link **before** the due date and time. The file name for the archive must have your student number as part of the file name, e.g. **uXXXXXXXXX.zip** (XXXXXXXXX is your student number)
2. The practical will be marked through a live demo on Discord.

**NO LATE** submissions will be accepted after the submission date and time has lapsed. Do not wait till the last minute to submit and start giving excuses that you faced technical challenges when you tried to submit.

# Question 1: PostgreSQL ORDMS

## Problem statement (from Practical 3)

The Student Records Unit of a university needs to keep records of all **students: undergraduates** and **postgraduates**, as well as **degree programs**, and **courses**. For each **student** the student number, name (title, first name, surname), date of birth, degree program, and the year of study (1<sup>st</sup> year, 2<sup>nd</sup> year, etc), should be recorded. For an **undergraduate student**, the courses currently registered for, should additionally be recorded. For a **postgraduate student**, the category (part time or full time), and supervisor (title, first name, surname), should additionally be recorded. For each **degree program** the degree name, number of years, department, and faculty should be recorded. For each **course**, the course code, course name, course credits, and department should be recorded.

## Improvements to the database design

A simple database design was used in practical 3. In this practical exercise, the database design specification is extended as follows:

**A.** It is necessary to ensure that all data that is entered in the database is valid. While the DBMS can check values against built-in and user-defined data types, there are some aspects of data values the DBMS does not have the ability to check, unless it is provided with triggers and special (trigger) functions for this purpose.

It is necessary to ensure that:

- (1) each *courseCode* in an undergraduate student's *courseRegistration* array is a valid code which already exists in the Course table (referential integrity).
- (2) the *courseCode-s* in an undergraduate student's *courseRegistration* array cannot be duplicated.
- (3) each *degreeCode* for a student's record is a valid code which already exists in the DegreeProgram table (referential integrity).

Since *courseRegistration* is an array, referential integrity cannot be implemented using *foreign keys*. A straight forward object-relational database design for the *degree code* referential integrity constraint would be to implement the **Student** table with a *foreign key* that references a row in the **DegreeProgram** table (by specifying REFERENCES **DegreeProgram**). Unfortunately, in PostgreSQL 14.5 inheritance and foreign keys are not compatible. (refer to chapter 5 section 5.10 of the PostgreSQL 14.5 documentation for details). So, for the database implementation in this practical exercise we will use triggers to enforce the referential integrity between the tables **Student**, **Undergraduate**, **Postgraduate**, **DegreeProgram** and **Course**.

**B.** It is also necessary to record the details of all the database users who perform DELETE operations on the **Undergraduate** and **Postgraduate** tables as well as the data they have deleted.

The following table summarises the above aspects of the required database design for this practical exercise. Some of these aspects are similar to practical 2.

Database 'object' type	Database 'objects'	Description
SEQUENCE (same as Prac3)	degreeSeq	generates surrogate (primary) keys for DegreeProgram table
	courseSeq	generates surrogate (primary) keys for the Course table
	studentSeq	generates surrogate (primary) keys for the Student, Undergraduate and Postgraduate tables
DOMAIN (same as Prac3)	StudentNumType	specifies that student number is six characters
	StudyYearsType	specifies that the number of years of study is one (or two) integer digits
TYPE (same as Prac3)	CategoryType	description of postgraduate categories with values: Part_time, Full_time
	TitleType	description of titles with values: Ms, Mev, Miss, Mrs, Mr, Mnr, Dr, Prof
	NameType	ROW type holding (title, firstName, surname)
TABLE (same as Prac3)	DegreeProgram	with attributes: degreeKey, degreeCode, degreeName , numberOfYears,faculty
	Course	with attributes: courseKey, courseCode, courseName, department, credits
	Student	with attributes: studentKey, studentNumber, fullName, DateOfBirth, degreeCode, yearOfStudy
	Undergraduate	inherits Student and additionally has attribute: courseRegistration
	Postgraduate	inherits Student and additionally has attributes: category, supervisor
These are new tables	DeletedUndergrad	When an Undergraduate student record is deleted, the deleted record is copied to this table. Attributes are: (all the attributes of <b>Undergraduate</b> , plus date-and-time of deletion, userid of the user who deleted the record)
	DeletedPostgrad	When a Postgraduate student record is deleted, the deleted record is copied to this table. Attributes are: (all the attributes of <b>Postgraduate</b> , plus date-and-time of deletion, userid of the user who deleted the record)
FUNCTION (you must create PL/pgSQL functions)	personFullNames (NameType)	returns a text string with the title, first name and surname
	ageInYears (DATE)	returns a student's age in years
	isRegisteredFor (text, text[ ])	returns a Boolean value to indicate if the undergraduate student is registered for the course with code passed in the first parameter.
	isValidCourseCode	returns true if the given courseCode value is one of the course codes in the Course table.
	hasValidCourseCodes	returns true if the new record to be inserted into the Undergraduate table has valid course codes which exist in the Course table.
	courseCodeFrequency	returns the frequency of a course code in a courseRegistration array

	hasDuplicateCourseCodes	returns true if the new record to be inserted into the Undergraduate table has duplicate course codes. <b>HINT: use a FOREACH loop within a FOREACH loop.</b>
	isValidDegreeCode	returns true if the new record to be inserted into the Undergraduate table has a valid degree code which exists in the DegreeProgram table.
TRIGGER PROCEDURE (you must create PL/pgSQL functions)	check_valid_degree_code	Trigger procedure called by <i>check_valid_degree</i> . Return type is TRIGGER.
	check_valid_course_codes	Trigger procedure called by <i>check_valid_course_registration</i> . Return type is TRIGGER. Checks that all course codes are valid and there are no duplicates.
	record_delete_undergrad, record_delete_postgrad,	Trigger procedures called by <i>audit_delete_undergrad</i> , <i>audit_delete_postgrad</i> . Return type is TRIGGER. <i>record_delete_undergrad</i> writes data for the deleted undergraduate in the <b>DeletedUndergrad</b> table and <i>record_delete_postgrad</i> writes data for the deleted Postgraduate in the <b>DeletedPostgrad</b> table.
TRIGGERS	check_valid_degree	Triggers on the Student, Undergraduate and Postgraduate tables for INSERT & UPDATE operations. Checks that the degreeCode is valid. Uses the <i>check_valid_degree_code</i> function.
	check_valid_course_registration	Trigger on the Undergraduate table for INSERT & UPDATE operations. Checks the course codes in the CourseRegistration array for an undergraduate. Uses the <i>check_valid_course_codes</i> function.
	audit_delete_undergrad audit_delete_postgrad	Triggers on the Undergraduate and Postgraduate tables for DELETE operations. Uses functions <i>record_delete_undergrad</i> and <i>record_delete_postgrad</i>

## Task 1: Creation of the database ‘objects’

[30 marks]

- Using PostgreSQL, write SQL statements and PL/pgSQL code to create all the database ‘objects’ in the above table and any other ‘objects’ you consider to be necessary. **Note:** you should re-use the CREATE statements for practical 3 for creating domains, types, sequences and tables.
- Create a database in PostgreSQL called *StudentsDBprac3* and run all the SQL statements in (1) above to create the database ‘objects’.** **Note:** marks for part (2) will only be awarded if the database ‘objects’ actually get created. Marks will be awarded as follows:

a.	Domains, Sequences and types:	no marks
b.	Tables from practical 3:	no marks
c.	New tables	2 marks

d.	Re-implementation of	
	Practical 2 functions using PL/pgSQL:	4 marks
e.	Implementation of the new functions	10 marks
f.	Functions for triggers:	8 marks
g.	Triggers:	6 marks

## Task 2: Inserting data into the Database tables

[no marks]

Re-use the *INSERT INTO* SQL statements of practical 3 to add the following data (same as practical 3) into the database. Execute some *SELECT* statements to confirm that you entered the data correctly.

	<b>Attribute values:</b> note that the values of attributes ... <b>key</b> are generated by the SEQUENCES that you created								
<b>Degree Program</b>	<b>degree key</b>	<b>degree code</b>	<b>degree name</b>	<b>number of years</b>	<b>Faculty</b>				
		BSc	Bachelor of Science	3	EBIT				
		BIT	Bachelor of IT	4	EBIT				
		PhD	Philosophiae Doctor	5	EBIT				
<b>Course</b>	<b>course key</b>	<b>course code</b>	<b>course name</b>	<b>department</b>	<b>credits</b>				
		COS301	Software Engineering	Computer Science	40				
		COS326	Database Systems	Computer Science	20				
		MTH301	Discrete Mathematics	Mathematics	15				
		PHL301	Logical	Philosophy	15				
<b>Under graduate</b>	<b>student key</b>	<b>student number</b>	<b>student name (title, fname, surname)</b>	<b>date of birth (dd-mm-yyyy)</b>	<b>degree code</b>	<b>year of study</b>	<b>courseRegistration</b>		
		140010	choose title & names	10-01-1996	BSc	3	COS301, COS326, MTH301		
		140015	choose title & names	25-05-1995	BSc	3	COS301, PHL301, MTH301		
		131120	choose title & names	30-01-1995	BIT	3	COS301, COS326, PHL301		
		131140	choose title & names	20-02-1996	BIT	4	COS301, COS326, MTH301, PHL301		
<b>Postgraduate</b>	<b>student key</b>	<b>student number</b>	<b>student name (title, fname, surname)</b>	<b>date of birth</b>	<b>degree code</b>	<b>year of study</b>	<b>category</b>	<b>supervisor (title,fname, surname)</b>	
		101122	choose title & names	15-06-1987	PhD	2	full time	choose title & names	
		121101	choose title & names	27-04-1985	PhD	3	part time	choose title & names	

### Task 3: SELECT statements to test the functions (10 marks)

1. Write a SELECT statement that will list the following details of all students who are registered for COS326: studentNumber, full names, age, degree program. The query must use the functions: *personFullNames*, *ageinYears*, *isRegisteredFor*. (2 marks)
2. Write two SELECT statements to demonstrate that the *hasValidCourseCodes* function provides the correct result. (4 marks)
3. Write two SELECT statements to demonstrate that the *hasDuplicateCourseCodes* function provides the correct result. (4 marks)

### Task 4: Insert, update and delete operations to test the triggers (10 marks)

Write and test SQL statements to do the operations listed in the table below. Marks will only be awarded if the SQL statement produces the correct output.

Operation	Table	values / actions	Testing trigger (for)	Marks
INSERT	Undergraduate	a record with an invalid degree code	<i>check_valid_degree</i> (referential integrity)	1
INSERT	Undergraduate	a record with a courseRegistration array which has an invalid course code	<i>check_valid_course_registration</i> (referential integrity)	1
INSERT	Postgraduate	a record with an invalid degree code	<i>check_valid_degree</i> (referential integrity)	1
UPDATE	Undergraduate	to change the degree code of an existing record to an invalid degree code	<i>check_valid_degree</i> (referential integrity)	1
UPDATE	Undergraduate	to change the CourseRegistration array of an existing record to another array which has an invalid course code	<i>check_valid_course_registration</i> (referential integrity)	1
UPDATE	Postgraduate	to change the degree code of an existing record to an invalid degree code	<i>check_valid_degree</i> (referential integrity)	1
DELETE	Undergraduate	delete one existing record	<i>audit_delete_undergrad</i> (auditing)	2
DELETE	Undergraduate	delete one existing record	<i>audit_delete_postgrad</i> (auditing)	2