МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБО6РОСТРОЕНИЯ»

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| старший преподаватель |  |  |  | Н.В. Путилова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| «Хранимые процедуры» |
| по курсу: Проектирование баз данных |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134К |  |  |  | Самарин Д. В. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2023

1. **Цель работы:**

Написать хранимые процедуры

1. **Задание на лабораторную работу:**

создать в БД хранимые процедуры, реализующие:  
— вставку с пополнением справочников (получаем ссылку на внешний ключ по значению данных из родительской таблицы, если данных нет- добавляем в родительскую, затем вставляем в дочернюю);  
— удаление с очисткой справочников – удаление данных из родительской таблицы, если после удаления данных из дочерней у строки родительской больше нет зависимых (удаляется информация о студенте, если в его группе нет больше студентов, запись удаляется из таблицы с перечнем групп);  
— каскадное удаление (удаление всех зависимых данных);  
— вычисление и возврат значения агрегатной функции (т.к. агрегатная функция дает единственный результат) ( задача- вернуть данные из процедуры/функции);  
— формирование статистики во временной таблице. (задача- работа с временными таблицами).

**Вариант задания (16 вариант)**

Школьные экскурсии: тип (развлекательная/образовательная), дисциплины к которым имеет отношение образовательная экскурсия, стоимость с человека, список участников, ответственный за проведение учитель

а. экскурсии на верфи (слово «верфь» в любом месте названия)

б. экскурсии, относящиеся к биологии и географии

в. учителя, не отвечающие за экскурсии

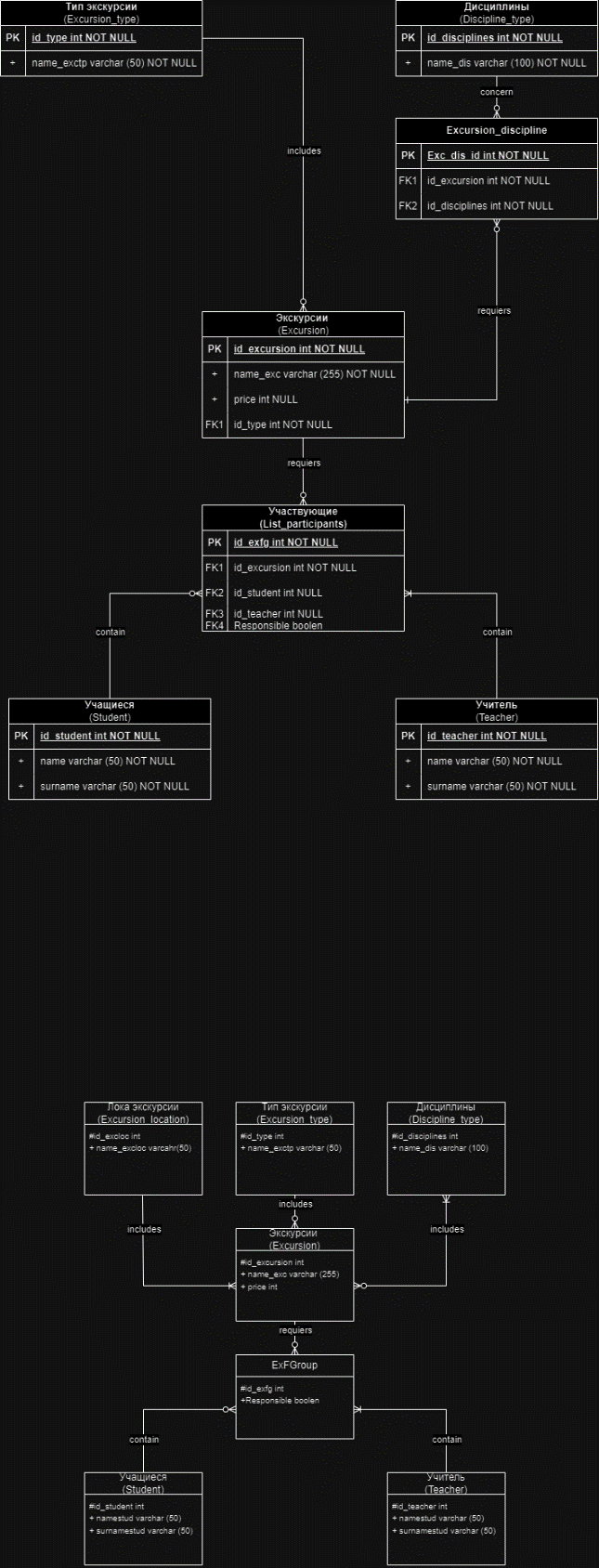
г. учащиеся, которые не ездили в музей истории религии, но ездили в Эрмитаж

д. учитель, отвечавший за число экскурсий больше среднего

е. учителя, отвечающие за самые дорогие экскурсии

ж. Экскурсии, на которые ездили все учащиеся

**3. Физическая модель БД**



**4.**

**Запросы к пунктам:**

Вставку с пополнением справочников (получаем ссылку на внешний ключ по значению данных из родительской таблицы, если данных нет- добавляем в родительскую, затем вставляем в дочернюю);

|  |
| --- |
| -- определим процедуру --  CREATE OR REPLACE PROCEDURE AddNewStudent(  firstName VARCHAR(50),  lastName VARCHAR(50),  tripName VARCHAR(255),  tripCost INTEGER,  typeId INTEGER  )  LANGUAGE plpgsql  AS $$  DECLARE  studentId INTEGER;  excursionId INTEGER;  participantId INTEGER;  BEGIN  -- Добавление нового студента, если он не существует  IF NOT EXISTS (SELECT 1 FROM student WHERE student\_name = firstName AND student\_secondname = lastName)  THEN  SELECT COALESCE(MAX(student\_id), 0) + 1 INTO studentId FROM student;  INSERT INTO student(student\_id, student\_name, student\_secondname)  VALUES (studentId, firstName, lastName);  ELSE  SELECT student\_id INTO studentId FROM student  WHERE student\_name = firstName AND student\_secondname = lastName;  END IF;  -- Добавление новой экскурсии, если она не существует  IF NOT EXISTS (SELECT 1 FROM excursion WHERE excursion\_name = tripName AND price = tripCost AND excursion\_type\_id = typeId)  THEN  SELECT COALESCE(MAX(excursion\_id), 0) + 1 INTO excursionId FROM excursion;  INSERT INTO excursion(excursion\_id, excursion\_name, price, excursion\_type\_id)  VALUES (excursionId, tripName, tripCost, typeId);  ELSE  SELECT excursion\_id INTO excursionId FROM excursion  WHERE excursion\_name = tripName AND price = tripCost AND excursion\_type\_id = typeId;  END IF;  -- Добавление в список участников  IF NOT EXISTS (SELECT 1 FROM list\_of\_participants WHERE student\_id = studentId AND excursion\_id = excursionId)  THEN  SELECT COALESCE(MAX(list\_of\_particip\_id), 0) + 1 INTO participantId FROM list\_of\_participants;  INSERT INTO list\_of\_participants(list\_of\_particip\_id, excursion\_id, student\_id)  VALUES (participantId, excursionId, studentId);  END IF;  END;  $$;  -- Вызов процедуры  CALL AddNewStudent('Иванов', 'Иван', 'Музей ракетостроения', 500, 2); |

Удаление с очисткой справочников – удаление данных из родительской таблицы, если после удаления данных из дочерней у строки родительской больше нет зависимых (удаляется информация о студенте, если в его группе нет больше студентов, запись удаляется из таблицы с перечнем групп);

|  |
| --- |
| CREATE OR REPLACE PROCEDURE ClearParticipantReference(  participant\_id INT  )  LANGUAGE plpgsql  AS $$  DECLARE  associated\_student\_id INT;  BEGIN  SELECT student\_id INTO associated\_student\_id FROM list\_of\_participants WHERE list\_of\_particip\_id = participant\_id;  DELETE FROM list\_of\_participants WHERE list\_of\_particip\_id = participant\_id;  IF NOT EXISTS (SELECT 1 FROM list\_of\_participants WHERE student\_id = associated\_student\_id) THEN  DELETE FROM student WHERE student\_id = associated\_student\_id;  END IF;  END;  $$;  -- Вызов процедуры  CALL ClearParticipantReference(41); |

Каскадное удаление (удаление всех зависимых данных);

|  |
| --- |
| -- Удаление экскурсии и связанных записей  CREATE OR REPLACE PROCEDURE removeExcursionWithDependencies(  targetExcursionId INT  )  LANGUAGE plpgsql  AS $$  BEGIN  -- Удаляем записи из таблицы excursion\_discipline, связанные с экскурсией  DELETE FROM excursion\_discipline WHERE excursion\_id = targetExcursionId;  -- Удаляем записи из таблицы list\_of\_participants, связанные с экскурсией  DELETE FROM list\_of\_participants WHERE excursion\_id = targetExcursionId;    -- Удаляем экскурсию из таблицы excursion  DELETE FROM excursion WHERE excursion\_id = targetExcursionId;  END;  $$;  -- Вызов процедуры удаления  CALL removeExcursionWithDependencies(8); |

Вычисление и возврат значения агрегатной функции (т.к. агрегатная функция дает единственный результат) (задача- вернуть данные из процедуры/функции);

|  |
| --- |
| -- Вычисление средней стоимости экскурсий  CREATE OR REPLACE FUNCTION CalculateAvgExcursionCost()  RETURNS integer AS $$  DECLARE  avgCost integer;  BEGIN  SELECT AVG(price) INTO avgCost FROM excursion;  RETURN avgCost;  END;  $$ LANGUAGE plpgsql;  -- Вызов функции для получения средней стоимости  SELECT CalculateAvgExcursionCost(); |

Формирование статистики во временной таблице. (задача- работа с временными таблицами).

|  |
| --- |
| CREATE OR REPLACE FUNCTION CalculatePriceVariationForExcursions()  RETURNS TABLE(excursion\_id INT, difference\_in\_price NUMERIC(10,2))  LANGUAGE plpgsql AS $$  DECLARE  averagePrice NUMERIC(10,2);  BEGIN  -- Создаем временную таблицу для анализа цен  CREATE TEMP TABLE IF NOT EXISTS TempPriceAnalysis (  id INT,  price NUMERIC(10,2)  );  -- Заполняем временную таблицу данными о ценах экскурсий  INSERT INTO TempPriceAnalysis (id, price)  SELECT  ex.excursion\_id,  ex.price  FROM  excursion ex;  -- Рассчитываем среднюю цену экскурсий  SELECT AVG(price) INTO averagePrice FROM TempPriceAnalysis;  -- Возвращаем разницу между ценой каждой экскурсии и средней ценой  RETURN QUERY SELECT  ex.excursion\_id,  ex.price - averagePrice AS difference\_in\_price  FROM  excursion ex;  -- Удаляем временную таблицу после использования  DROP TABLE IF EXISTS TempPriceAnalysis;  END;  $$;  -- Вызов функции и вывод результатов  SELECT \* FROM CalculatePriceVariationForExcursions(); |

**Вывод:** выполнив лабораторную работу я освоил навыки работы с хранимыми процедурами