Федеральное государственное автономное образовательное учреждение

высшего образования «Санкт-Петербургский государственный университет аэрокосмического приборостроения» (ГУАП)

ОТЧЕТ ЗАЩИЩЕН

Ст. преподаватель Путилова Н.В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Должность Ф.И.О дата, подпись

**Отчет**

**о лабораторной работе № 4**

**«Объектно-реляционные базы данных.**

**Манипуляция данными и пользовательские операторы»**

по дисциплине «Проектирование баз данных»

ОТЧЕТ ВЫПОЛНИЛ:

Студент Самарин Д.В. 4134 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ф.И.О. группа дата, подпись

Санкт-Петербург

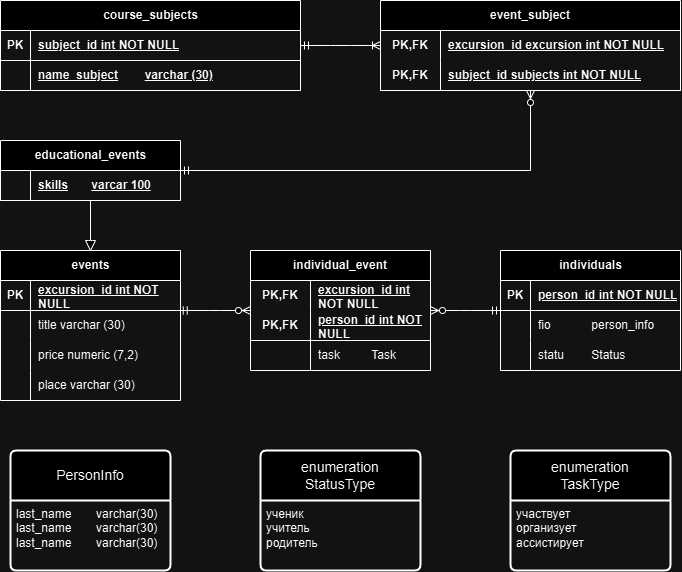
2024

**Вариант 16**

Школьные экскурсии: тип (развлекательная/образовательная), дисциплины к которым имеет отношение образовательная экскурсия, стоимость с человека, список участников, ответственный за проведение учитель, категории участников экскурсии (участвует, организует, ассистирует при проведении (старшие классы у младших или родители))

1. экскурсии в усадьбы (слово усадьба в любом месте названия)
2. экскурсии, относящиеся к 2 дисциплинам
3. учащиеся, которые не ездили в музей истории религии
4. экскурсия, собравшая наибольшее число ассистентов
5. учителя, отвечающие за самые дорогие экскурсии

**Физическая модель БД**

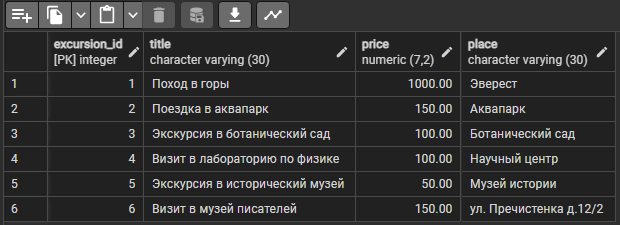


**Скрипт заполнения базы данных:**

|  |
| --- |
| *-- Заполнение таблицы subjects*  *INSERT INTO subjects (name\_subject) VALUES*  *('Математика'),*  *('Русский язык'),*  *('Литература'),*  *('История'),*  *('Физика');*  *-- Заполнение таблицы person*  *INSERT INTO person (fio, status) VALUES*  *(ROW('Сергеев', 'Николай', 'Васильевич'), 'учитель'),*  *(ROW('Михайлов', 'Дмитрий', 'Игоревич'), 'ученик'),*  *(ROW('Попова', 'Ольга', 'Викторовна'), 'родитель'),*  *(ROW('Васильева', 'Мария', 'Александровна'), 'учитель'),*  *(ROW('Козлов', 'Александр', 'Юрьевич'), 'ученик');*  *-- Заполнение таблицы entertainment\_excursion*  *INSERT INTO entertainment\_excursion (title, price, place) VALUES*  *('Поход в горы', 1000.00, 'Эверест'),*  *('Поездка в аквапарк', 150.00, 'Аквапарк'),*  *('Экскурсия в ботанический сад', 100.00, 'Ботанический сад');*  *-- Заполнение таблицы educational\_ex*  *INSERT INTO educational\_ex (title, price, place, skills) VALUES*  *('Визит в лабораторию по физике', 100.00, 'Научный центр', 'навыки экспериментальной работы и анализа данных'), --физика*  *('Экскурсия в исторический музей', 50.00, 'Музей истории', 'понимание исторических процессов и критическое мышление'), --история*  *('Визит в музей писателей', 150.00, 'ул. Пречистенка д.12/2', 'знание классической литературы и развитие литературного вкуса'); --литература*  *-- Заполнение таблицы ed\_sub*  *INSERT INTO ed\_sub (excursion\_id, subject\_id) VALUES*  *(4, 5), -- Экскурсия в лабораторию по физике, Физика*  *(5, 4), -- Экскурсия в исторический музей, История*  *(6, 3); -- Визит в музей писателей, Литература*  *-- Заполнение таблицы person\_excursion*  *INSERT INTO person\_excursion (excursion\_id, person\_id, task) VALUES*  *(1, 2, 'участвует'), -- Поход в горы, Михайлов, ученик*  *(2, 1, 'организует'), -- Поездка в аквапарк, Сергеев, учитель*  *(3, 5, 'ассистирует'), -- Экскурсия в ботанический сад, Козлов, ученик*  *(4, 3, 'участвует'), -- Визит в лабораторию по физике, Попова, родитель*  *(5, 4, 'участвует'),-- Экскурсия в исторический музей, Васильева, учитель*  *(5, 1, 'ассистирует'); -- Визит в музей писателей, Сергеев, учитель* |

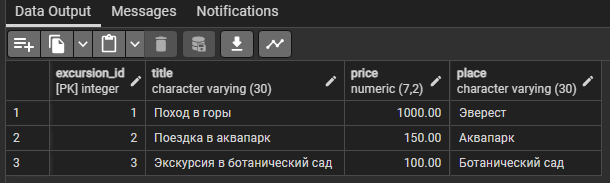
**Запрос к предку и потомку:**

|  |
| --- |
| SELECT \* FROM events; |

****

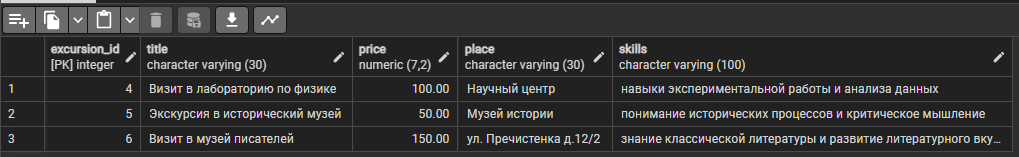
**Запрос только к предку:**

|  |
| --- |
| SELECT \* FROM ONLY events; |

****

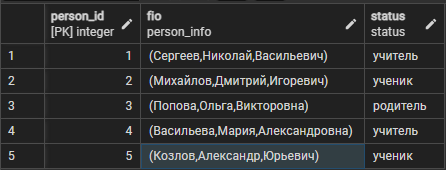
**Запрос только к потомку:**

|  |
| --- |
| SELECT \* FROM educational\_events; |

****

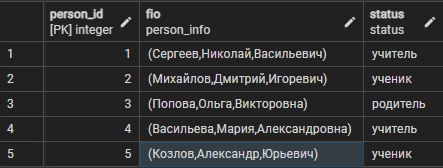
**Пользовательский оператор (сравнение по алфавиту, у кого фамилия (при равенстве имя или отчество) начинается с более ранней буквы, тот и выводится):**

|  |
| --- |
| *-- Пользовательский оператор*  *CREATE OR REPLACE FUNCTION compare\_person\_info(p1 PersonInfo, p2 PersonInfo) RETURNS PersonInfo AS $$*  *BEGIN*  *IF p1.last\_name < p2.last\_name THEN*  *RETURN p1;*  *ELSIF p1.last\_name > p2.last\_name THEN*  *RETURN p2;*  *ELSE*  *IF p1.name < p2.name THEN*  *RETURN p1;*  *ELSIF p1.name > p2.name THEN*  *RETURN p2;*  *ELSE*  *IF p1.patronymic < p2.patronymic THEN*  *RETURN p1;*  *ELSIF p1.patronymic > p2.patronymic THEN*  *RETURN p2;*  *ELSE*  *RETURN p1;*  *END IF;*  *END IF;*  *END IF;*  *END;*  *$$ LANGUAGE plpgsql;*  *CREATE OPERATOR ~~~ (*  *PROCEDURE = compare\_person\_info,*  *LEFTARG = PersonInfo,*  *RIGHTARG = PersonInfo --типы данных, которые принимаются слева и справа,*  *--в моем случае - одинаковые*  *);*  *-- Пример работы оператора*  *SELECT (p1.personal\_info ~~~ p2.personal\_info) AS comparison\_result*  *FROM individuals p1, individuals p2*  *WHERE p1.individual\_id = 3 AND p2.individual\_id = 5;* |
| **Вывод:** |

****

**Пользовательская агрегатная функция (выводит первого по алфавиту человека):**

|  |
| --- |
| CREATE OR REPLACE FUNCTION sort\_people\_agg(p1 PersonInfo, p2 PersonInfo) RETURNS PersonInfo AS $$  BEGIN  IF (p1 ~~~ p2) = p1 THEN  RETURN p1;  ELSE  RETURN p2;  END IF;  END;  $$ LANGUAGE plpgsql;  -- Пользовательская агрегатная функция  CREATE AGGREGATE sort\_people(  BASETYPE = PersonInfo, -- Базовый тип данных для агрегации  SFUNC = sort\_people\_agg, -- Функция состояния  STYPE = PersonInfo -- Тип данных, который возвращает агрегатная функция (тип состояния)  );  -- Пример работы агрегатной функции:  SELECT sort\_people(personal\_info) FROM individuals; |
| **Вывод:** |

****