МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

Национальный исследовательский ядерный университет «МИФИ»

Институт
интеллектуальных кибернетических систем

Кафедра кибернетики (№ 22)

Отчет о работе по курсу

«Базы данных (теоретические основы баз данных)»

Выполнил	Колесников М. Л.
Группа	Б22-534
Вариант	Расписание и сдача ЕГЭ
Преподаватель	Петровская А.В.
Проверяющий	
Оценка	

Москва, 2021

Формулировка задания

Спроектировать базу данных для проведения Единого Государственного экзамена, проводящегося ежегодно в школах разных городов Российской Федерации. База данных должна содержать информацию о студентах, школах и учителях, а также отражать ежегодные данные по сдаваемым предметам, составленное расписание и полученные учениками результаты.

Концептуальная модель базы данных

После проведения анализа предметной области была спроектирована следующая концептуальная модель:

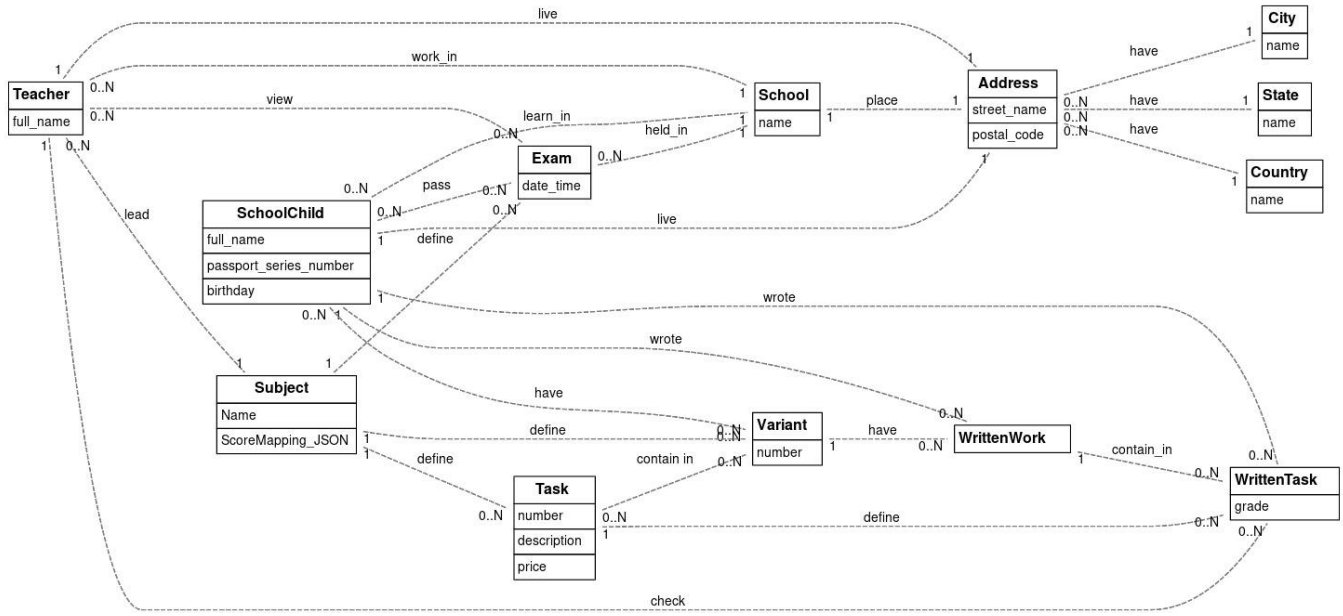

Рисунок 1 – Концептуальная модель базы данных

Конкретизация предметной области

Необходимо создать систему, отражающую информацию о проведении и результатах экзаменов по всей стране. По каждому предмету есть ежегодная информация, так как Министерство образования ежегодно вносит коррективы в тот или иной экзамен. База данных должна отражать точное расписание экзаменов по всем городам каждый год, а также результаты конкретного ученика по всем выбранным им предметам.

Описание предметной области

Система рассчитана на работу с зарегистрированными представителями ФИПИ, а также людьми, принимающими непосредственное участие в составлении

расписания ЕГЭ. Обычные пользователи: ученики, родители учеников, учителя и т.д. доступа к этой базе данных не имеют.

Каждому конкретному ученику и родителю ученика в его личном кабинете доступна следующая информация: расписание проведения выбранных экзаменов, а также в дальнейшем результаты по каждому экзамену, включающие в себя распределение баллов по каждому заданию, количество первичных баллов и количество баллов по стобалльной шкале. В школу же (лично администрации данной школы) после объявления результатов по каждому конкретному экзамену приходят сведения по всем ученикам данной школы с их результатами. Затем по усмотрению администрации данная информация распространяется по классным руководителям и, наконец, доводится до учеников.

Каждый экзамен проводится в школе, где на каждую аудиторию, вмещающую до 18 человек назначен наблюдатель. Каждому ученику, сдающему конкретный экзамен в конкретной школе выделена аудитория и персональное место в этой аудитории.

Описание атрибутов

В процессе анализа были выделены следующие атрибуты, название и описание которых приведены в таблице ниже:

Имя атрибута	Расшифровка
id	Уникальный идентификатор. Есть у каждого объекта.
Имя, фамилия, Отчество	Имя, фамилия, отчество ученика, принимающего участие в экзаменах или учителя, который является наблюдателем на экзамене
Серия и номер паспорта	Документом, удостоверяющим личность для допуска на экзамен, является русский паспорт
Дата рождения	Дата рождения ученика.
Название предмета	Название предмета, экзамен по которому сдают ученики
Название школы	Название школы, где проводится экзамен.
Адрес: Россия(страна), субъект, город, улица	Адрес местонахождения школы или адрес места жительства ученика.
Распределение баллов по заданиям	Показывает максимальный балл за каждое задание по данному предмету.

Логическое проектирование

Следующим шагом на с помощью dbdiagram.io была разработана логическая модель базы данных, имеющая нормальную форму, представленная ниже:

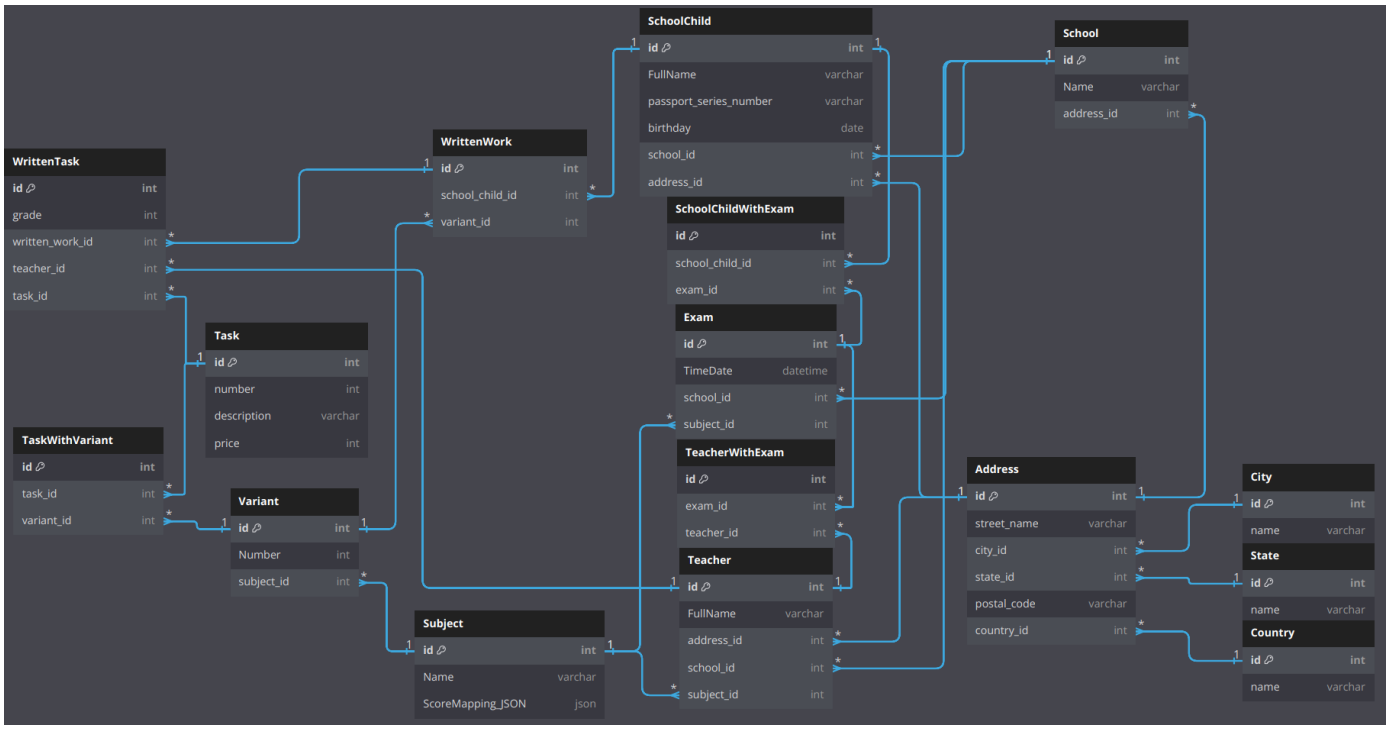

Рисунок 2 - Логическая модель базы данных

Физическое проектирование

В качестве СУБД для реализации разработано базы данных была выбрана PostgreSQL. В связи с проведенным анализом предметно области была проработана следующая физическая схема БД. Она представлена на следующем рисунке:

Рисунок 3 – Графическое представление базы данных

Создание таблиц

Ниже приведены запросы на языке SQL (диалект PostgreSQL) для создания таблиц, описанных выше:

```
CREATE TABLE IF NOT EXISTS City (
  id SERIAL PRIMARY KEY,
  name VARCHAR
);
CREATE TABLE IF NOT EXISTS State (
  id SERIAL PRIMARY KEY,
  name VARCHAR
);
CREATE TABLE IF NOT EXISTS Country (
  id SERIAL PRIMARY KEY,
  name VARCHAR
);
CREATE TABLE IF NOT EXISTS Address (
  id SERIAL PRIMARY KEY,
  street VARCHAR,
  city_id INT,
  state_id INT,
  postal_code VARCHAR,
  country_id INT,
  FOREIGN KEY (city_id) REFERENCES City(id),
  FOREIGN KEY (state_id) REFERENCES State(id),
  FOREIGN KEY (country_id) REFERENCES Country(id)
);
CREATE TABLE IF NOT EXISTS School (
  id SERIAL PRIMARY KEY,
```

```
    name VARCHAR,
    address_id INT,
    FOREIGN KEY (address_id) REFERENCES Address(id)
);
CREATE TABLE IF NOT EXISTS Subject (
    id SERIAL PRIMARY KEY,
    name VARCHAR,
    ScoreMapping_JSON JSON
);
CREATE TABLE IF NOT EXISTS SchoolChild (
    id SERIAL PRIMARY KEY,
    FullName VARCHAR,
    passport_series_number VARCHAR UNIQUE,
    birthday DATE,
    school_id INT,
    address_id INT,
    FOREIGN KEY (school_id) REFERENCES School(id),
    FOREIGN KEY (address_id) REFERENCES Address(id)
);
CREATE TABLE IF NOT EXISTS Exam (
    id SERIAL PRIMARY KEY,
    TimeDate TIMESTAMP,
    school_id INT,
    subject_id INT,
    FOREIGN KEY (school_id) REFERENCES School(id),
    FOREIGN KEY (subject_id) REFERENCES Subject(id)
);
CREATE TABLE IF NOT EXISTS SchoolChildWithExam (
```

```
    id SERIAL PRIMARY KEY,
    school_child_id INT,
    exam_id INT,
    FOREIGN KEY (school_child_id) REFERENCES SchoolChild(id),
    FOREIGN KEY (exam_id) REFERENCES Exam(id)
);
CREATE TABLE IF NOT EXISTS Teacher (
    id SERIAL PRIMARY KEY,
    FullName VARCHAR,
    address_id INT,
    school_id INT,
    subject_id INT,
    FOREIGN KEY (address_id) REFERENCES Address(id),
    FOREIGN KEY (school_id) REFERENCES School(id),
    FOREIGN KEY (subject_id) REFERENCES Subject(id)
);
CREATE TABLE IF NOT EXISTS TeacherWithExam (
    id SERIAL PRIMARY KEY,
    exam_id INT,
    teacher_id INT,
    FOREIGN KEY (exam_id) REFERENCES Exam(id),
    FOREIGN KEY (teacher_id) REFERENCES Teacher(id)
);
CREATE TABLE IF NOT EXISTS Variant (
    id SERIAL PRIMARY KEY,
    number INT CHECK (number >= 1),
    subject_id INT,
    FOREIGN KEY (subject_id) REFERENCES Subject(id)
```

```
);
CREATE TABLE IF NOT EXISTS Task (
  id SERIAL PRIMARY KEY,
  number INT CHECK (number >= 1),
  description VARCHAR,
  price INT CHECK (price >= 1)
);
CREATE TABLE IF NOT EXISTS TaskWithVariant (
  id SERIAL PRIMARY KEY,
  task_id INT,
  variant_id INT,
  FOREIGN KEY (task_id) REFERENCES Task(id),
  FOREIGN KEY (variant_id) REFERENCES Variant(id)
);
CREATE TABLE IF NOT EXISTS WrittenWork (
  id SERIAL PRIMARY KEY,
  school_child_id INT,
  variant_id INT,
  FOREIGN KEY (school_child_id) REFERENCES SchoolChild(id),
  FOREIGN KEY (variant_id) REFERENCES Variant(id)
);
CREATE TABLE IF NOT EXISTS WrittenTask (
  id SERIAL PRIMARY KEY,
  grade INT CHECK (grade >= 0 AND grade <= 100),
  written_work_id INT,
  teacher_id INT,
  task_id INT,
  FOREIGN KEY (written_work_id) REFERENCES WrittenWork(id),
```

```
    FOREIGN KEY (teacher_id) REFERENCES Teacher(id),
    FOREIGN KEY (task_id) REFERENCES Task(id)
);
```

Заполнение базы данных

Заполнение базы данных реализовано при помощи языка программирования Python. Использовались библиотеки psycopg2 для работы с PostgreSQL, multiprocessing для разделения генерации данных по разным процессах с целью ускорить генерацию столь большого объема данных.

Подготовка данных

Написан парсер для подготовки данных о соответствии городов и субъектов РФ. По каждому предмету из экзамена подготовлены данные о максимально возможном количестве баллов за каждое задание. Реализованы функции генерации данных о школах, экзаменах, учителях, учениках и их адресах, о предметах.

Программа заполнения базы данных

https://github.com/MichaelKolesnikov/DataGenerator

Результаты заполнения

Далее представлены результаты работы программы на примере таблиц, соответствующих функциям, приведенным выше.

```
1  select * from schoolchild;
```

Data Output | Messages | Notifications

	id [PK] integer	fullname character varying	passport_series_number character varying	birthday date	school_id integer	address_id integer
1	1	Анжела Тимуровна Артемьева	0000000001	1929-02-24	5750	1040000
2	120001	Анжела Тимуровна Артемьева	0000120001	1929-02-24	18594	1160000
3	240001	Анжела Тимуровна Артемьева	0000240001	1929-02-24	3560	1280000
4	360001	Анжела Тимуровна Артемьева	0000360001	1929-02-24	33360	1400000
5	480001	Анжела Тимуровна Артемьева	0000480001	1929-02-24	29211	1520000
6	2	Лукина Анжелика Степановна	0000000002	2020-03-23	39315	1040001
7	3	Валентина Вениаминовна Ермакова	0000000003	1992-10-08	11642	1040002
8	4	Казаков Мартын Брониславович	0000000004	1990-02-05	33513	1040003
9	5	Туров Владимир Тарасович	0000000005	1986-03-18	33763	1040004
10	6	Поликарп Изотович Кулагин	0000000006	1995-09-20	6359	1040005
11	7	Суханов Ефим Ильич	0000000007	1929-01-12	33092	1040006
12	8	Доронин Виталий Захарьевич	0000000008	1935-02-11	16534	1040007
13	9	Прохорова Олимпиада Георгиевна	0000000009	1952-03-04	27966	1040008
14	10	Герасим Анисимович Кононов	0000000010	2004-08-01	25968	1040009
15	11	Петухова Анжелика Тимуровна	0000000011	1984-04-07	38842	1040010
16	12	Яковлева Ираида Степановна	0000000012	1956-12-13	10735	1040011
17	13	Костина Синклитикия Архиповна	0000000013	1979-07-23	28434	1040012
18	14	Рябов Ратибор Тарасович	0000000014	1929-09-13	20941	1040013
19	15	Кирилл Феоктистович Соколов	0000000015	1953-10-29	33626	1040014
20	16	Захарова Олимпиада Наумовна	0000000016	1970-05-24	7216	1040015
21	17	Ярополк Данилович Федотов	0000000017	1909-12-26	25543	1040016
22	18	Нестор Эдгарович Овчинников	0000000018	1961-06-22	5229	1040017
23	19	Герасимова Ираида Геннадиевна	0000000019	1999-12-25	29776	1040018
24	20	Ксения Анатольевна Гришина	0000000020	1990-04-05	5820	1040019

Total rows: 600000 | Query complete 00:00:00.895

```
1  select * from teacher;
```

Data Output | Messages | Notifications

	id [PK] integer	fullname character varying	address_id integer	school_id integer	subject_id integer
1	0	Анжела Тимуровна Артемьева	40000	24141	9
2	200000	Анжела Тимуровна Артемьева	240000	15934	12
3	400000	Анжела Тимуровна Артемьева	440000	38951	4
4	600000	Анжела Тимуровна Артемьева	640000	39092	10
5	800000	Анжела Тимуровна Артемьева	840000	17330	2
6	1	Вероника Васильевна Федосеева	40001	1431	13
7	2	Василиса Леоновна Кондратьева	40002	13465	7
8	3	Колобов Софон Аверьянович	40003	37671	6
9	4	Фадей Афанасьевич Кузьмин	40004	2231	1
10	5	Соломон Ефстафьевич Абрамов	40005	16000	7
11	6	Евсеев Христофор Иосифович	40006	30065	11
12	7	Вероника Архиповна Субботина	40007	22516	7
13	8	Гурьев Демьян Брониславович	40008	39524	1
14	9	Евдокия Максимовна Белоусова	40009	16132	10
15	10	Герасим Анисимович Кононов	40010	7493	9
16	11	Петухова Анжелика Тимуровна	40011	14640	2
17	12	Яковлева Ираида Степановна	40012	22682	7
18	13	Костина Синклитикия Архиповна	40013	17090	13
19	14	Григорьева Ангелина Максимовна	40014	10814	3
20	15	Владислав Жоресович Богданов	40015	4729	6
21	16	Тимофеев Алексей Геннадиевич	40016	5969	7
22	17	Борислав Захарьевич Беляев	40017	6967	3
23	18	Любосмысл Елизарович Гордеев	40018	9852	7
24	19	Никифорова Иванна Тимуровна	40019	5866	13

Total rows: 1000000 | Query complete 00:00:01.395

```
1  select * from exam;
```

Data Output | Messages | Notifications

	id [PK] integer	timedate timestamp without time zone	school_id integer	subject_id integer
1	384000	2024-06-09 22:16:20.307782	24000	0
2	384001	2024-01-14 08:51:18.92145	24000	1
3	384002	2024-05-18 13:19:46.05015	24000	2
4	384003	2024-04-23 15:58:38.39893	24000	3
5	384004	2024-10-20 01:45:22.9855	24000	4
6	384005	2023-12-17 10:32:24.203116	24000	5
7	384006	2024-05-03 10:07:41.967029	24000	6
8	384007	2024-04-10 19:04:44.212697	24000	7
9	384008	2024-06-20 03:03:55.262255	24000	8
10	384009	2024-02-03 01:16:28.768508	24000	9
11	384010	2024-03-31 21:14:25.183108	24000	10
12	384011	2024-01-16 10:35:29.339438	24000	11
13	384012	2024-02-10 17:48:27.713239	24000	12
14	384013	2024-02-14 07:12:14.617814	24000	13
15	384014	2024-05-28 03:11:45.909235	24000	14
16	384015	2024-10-14 17:14:32.710799	24000	15
17	384016	2024-10-10 19:32:02.117156	24001	0
18	384017	2024-09-21 04:40:38.462383	24001	1
19	384018	2024-08-06 23:12:36.546312	24001	2
20	384019	2024-06-13 10:09:03.836829	24001	3
21	384020	2024-06-05 00:24:33.52489	24001	4
22	384021	2024-04-29 08:07:06.950656	24001	5
23	384022	2024-09-05 22:34:08.085207	24001	6
24	384023	2024-10-24 01:43:38.372979	24001	7

Total rows: 640000 | Query complete 00:00:00.609

```
1  select * from subject;
```

Data Output | Messages | Notifications

	id [PK] integer	name character varying	scoremapping_json json
1	0	Математика. Базовый уровень	{"1": "1", "2": "1", "3": "1", "4": "1", "5": "1", "6": "1", "7": "1", "8": "1", "9": "1", "10": "1", "11": "1", "12": "1", "13": "1", "14"
2	1	Математика. Профильный уровень	{"1": "1", "2": "1", "3": "1", "4": "1", "5": "1", "6": "1", "7": "1", "8": "1", "9": "1", "10": "1", "11": "1", "12": "1", "13": "2", "14"
3	2	Русский язык	{"1": "1", "2": "1", "3": "1", "4": "1", "5": "1", "6": "1", "7": "1", "8": "2", "9": "1", "10": "1", "11": "1", "12": "1", "13": "1", "14"
4	3	Информатика и ИКТ	{"1": "1", "2": "1", "3": "1", "4": "1", "5": "1", "6": "1", "7": "1", "8": "1", "9": "1", "10": "1", "11": "1", "12": "1", "13": "1", "14"
5	4	Литература	{"1": "1", "2": "1", "3": "1", "4": "4", "5": "8", "6": "1", "7": "1", "8": "1", "9": "4", "10": "8", "11": "18"}
6	5	История	{"1": "2", "2": "1", "3": "2", "4": "3", "5": "2", "6": "2", "7": "2", "8": "1", "9": "1", "10": "1", "11": "1", "12": "2", "13": "2", "14"
7	6	Обществознание	{"1": "1", "2": "2", "3": "1", "4": "2", "5": "2", "6": "2", "7": "2", "8": "2", "9": "1", "10": "2", "11": "2", "12": "1", "13": "2", "14"
8	7	Английский язык	{"1": "2", "2": "3", "3": "1", "4": "1", "5": "1", "6": "1", "7": "1", "8": "1", "9": "1", "10": "3", "11": "2", "12": "1", "13": "1", "14"
9	8	Немецкий язык	{"1": "2", "2": "3", "3": "1", "4": "1", "5": "1", "6": "1", "7": "1", "8": "1", "9": "1", "10": "3", "11": "2", "12": "1", "13": "1", "14"
10	9	Французский язык	{"1": "2", "2": "3", "3": "1", "4": "1", "5": "1", "6": "1", "7": "1", "8": "1", "9": "1", "10": "3", "11": "2", "12": "1", "13": "1", "14"
11	10	Испанский язык	{"1": "2", "2": "3", "3": "1", "4": "1", "5": "1", "6": "1", "7": "1", "8": "1", "9": "1", "10": "3", "11": "2", "12": "1", "13": "1", "14"
12	11	Китайский язык	{"1": "6", "2": "1", "3": "1", "4": "1", "5": "1", "6": "1", "7": "1", "8": "1", "9": "1", "10": "6", "11": "4", "12": "1", "13": "1", "14"
13	12	Химия	{"1": "1", "2": "1", "3": "1", "4": "1", "5": "1", "6": "2", "7": "2", "8": "2", "9": "1", "10": "1", "11": "1", "12": "1", "13": "1", "14"
14	13	Биология	{"1": "1", "2": "2", "3": "1", "4": "1", "5": "1", "6": "2", "7": "2", "8": "2", "9": "1", "10": "2", "11": "2", "12": "2", "13": "1", "14"
15	14	Физика	{"1": "1", "2": "1", "3": "1", "4": "1", "5": "2", "6": "2", "7": "1", "8": "1", "9": "2", "10": "2", "11": "1", "12": "1", "13": "1", "14"
16	15	География	{"1": "1", "2": "1", "3": "1", "4": "1", "5": "2", "6": "1", "7": "1", "8": "1", "9": "1", "10": "1", "11": "1", "12": "2", "13": "1", "14"

Выполнение запросов

В этом разделе приведены различные запросы к реализованной базе данных — их краткие описания, непосредственно запрос на языке SQL и результат выполнения.

1. Простые запросы:
    a. Сколько вариантов подготовлено для каждого предмета

```
select subject.name, count(*) from subject join variant on variant.subject_id=subject.id group by subject.name;
```

Data Output | Messages | Notifications

	name character varying	count bigint
1	История	50
2	Биология	50
3	Китайский язык	50
4	Английский язык	50
5	Литература	50
6	Физика	50
7	География	50
8	Математика. Базовый уровень	50
9	Немецкий язык	50
10	Обществознание	50
11	Испанский язык	50
12	Русский язык	50
13	Химия	50
14	Математика. Профильный уровень	50
15	Французский язык	50
16	Информатика и ИКТ	50

    b. Вывести средний балл по каждому предмету для школьника с ID = 1

```
SELECT sub.name, AVG(wt.grade) AS average_grade
FROM WrittenTask wt
JOIN WrittenWork ww ON wt.written_work_id = ww.id
JOIN Variant v ON ww.variant_id = v.id
JOIN Subject sub ON v.subject_id = sub.id
WHERE ww.school_child_id = 1
GROUP BY sub.name;
```

Data Output | Messages | Notifications

	name character varying	average_grade numeric
1	Информатика и ИКТ	0.55555555555555555556
2	Математика. Базовый уровень	0.61904761904761904762
3	Русский язык	0.62962962962962962963

c. Найти все предметы, по которым проводились экзамены в школе с номером 1

```
SELECT sub.name
FROM Exam e
JOIN Subject sub on e.subject_id = sub.id
JOIN School s on e.school_id = s.id
WHERE s.name='Школа №1';
```

Data Output | Messages | Notifications

	name character varying
1	Математика. Базовый уровень
2	Математика. Профильный уровень
3	Русский язык
4	Информатика и ИКТ
5	Литература
6	История
7	Обществознание
8	Английский язык
9	Немецкий язык
10	Французский язык
11	Испанский язык
12	Китайский язык
13	Химия
14	Биология
15	Физика
16	География

d. Вывести топ-100 учителей по количеству проведенных экзаменов.

```sql
SELECT t.id, COUNT(te.exam_id) AS exam_count
FROM Teacher t
JOIN TeacherWithExam te ON t.id = te.teacher_id
GROUP BY t.id
ORDER BY exam_count DESC
LIMIT 10;
```

Data Output | Messages | Notifications

	id [PK] integer	exam_count bigint
1	108965	10
2	727759	10
3	589947	10
4	820751	10
5	829359	10
6	439741	10
7	478549	10
8	643138	10
9	472684	10
10	616008	10

2. Средние запросы:

a. Для каждого предмета вывести, сколько в нем заданий

```sql
SELECT
    name,
    (SELECT COUNT(*) FROM jsonb_object_keys(ScoreMapping_JSON::jsonb)) AS count_of_tasks
FROM subject;
```

Data Output | Messages | Notifications

	name character varying	count_of_tasks bigint
1	Математика. Базовый уровень	21
2	Математика. Профильный уровень	19
3	Русский язык	27
4	Информатика и ИКТ	27
5	Литература	11
6	История	21
7	Обществознание	25
8	Английский язык	42
9	Немецкий язык	42
10	Французский язык	42
11	Испанский язык	42
12	Китайский язык	32
13	Химия	34
14	Биология	28
15	Физика	26
16	География	29

b. Верно ли, что каждый учитель учит в школе, которая находится в том же городе, что и учитель?

```sql
with teacher_city as (
    select teacher.id, school_id, city_id from teacher join address on address.id=teacher.address_id
),
school_city as (
    select school.id as school_id, city_id from school join address on address.id=school.address_id
)
SELECT
    CASE
        WHEN EXISTS (
            SELECT 1
            FROM teacher_city
            JOIN school_city ON teacher_city.school_id = school_city.school_id
            WHERE teacher_city.city_id <> school_city.city_id
        ) THEN 'no'
        ELSE 'yes'
    END AS teacher_city_match
;
```

Data Output | Messages | Notifications

	teacher_city_match text
1	yes

c. Какой ребенок какой предмет насколько сдал

```sql
SELECT
    sc.id,
    s.name AS SubjectName,
    SUM(wt.grade) AS TotalScore
FROM
    SchoolChild sc
JOIN
    WrittenWork ww ON sc.id = ww.school_child_id
JOIN
    WrittenTask wt ON ww.id = wt.written_work_id
JOIN
    Task t ON wt.task_id = t.id
JOIN
    Variant v ON ww.variant_id = v.id
JOIN
    Subject s ON v.subject_id = s.id
GROUP BY
    sc.id, s.name
ORDER BY
    sc.id, s.name;
```

Data Output | Messages | Notifications

	id integer	subjectname character varying	totalscore bigint
1	1	Информатика и ИКТ	15
2	1	Математика. Базовый уровень	13
3	1	Русский язык	17
4	2	Китайский язык	36
5	2	Математика. Профильный уровень	11
6	2	Русский язык	23
7	3	Биология	34
8	3	Математика. Профильный уровень	21
9	3	Русский язык	28

Total rows: 1800000 | Query complete 00:00:54.400

d. Среднее количество учителей, живущих в одном городе.

```sql
SELECT AVG(teacher_count) AS average_teachers_per_city
FROM (
    SELECT city_id, COUNT(*) AS teacher_count
    FROM Teacher
    JOIN Address ON Teacher.address_id = Address.id
    GROUP BY city_id
) AS city_teacher_counts;
```

Data Output | Messages | Notifications

	average_teachers_per_city numeric
1	877.1929824561403509

3. Сложные запросы:

a. Для каждого учителя сравнить количество его проверенных заданий со средним количеством проверенных заданий одним учителем.

```sql
WITH teacher_tasks AS (
    SELECT
        teacher_id,
        COUNT(*) AS tasks_checked
    FROM writtentask
    GROUP BY teacher_id
)
SELECT
    teacher_id,
    tasks_checked,
    tasks_checked - AVG(tasks_checked) OVER () AS dif
FROM teacher_tasks
ORDER BY tasks_checked DESC;
```

Data Output | Messages | Notifications

	teacher_id integer	tasks_checked bigint	dif numeric
1	794757	341	294.2890207804156083
2	58161	328	281.2890207804156083
3	797243	324	277.2890207804156083
4	909970	324	277.2890207804156083
5	930090	324	277.2890207804156083
6	941537	322	275.2890207804156083
7	575464	322	275.2890207804156083
8	355173	322	275.2890207804156083
9	211934	321	274.2890207804156083
10	767323	321	274.2890207804156083
11	159798	321	274.2890207804156083
12	862565	321	274.2890207804156083
13	728933	320	273.2890207804156083
14	406100	319	272.2890207804156083
15	805048	319	272.2890207804156083

Total rows: 999980 | Query complete 00:00:31.368

b. Найти школьника, который получил самый высокий средний балл по всем предметам, вывести его id, средний балл.

```sql
create temporary table winner as (
    select 0 as id, 0 as average_grade
);
DO $$
DECLARE
    sub_id INTEGER;
    max_grade INTEGER;
    max_school_child_id INTEGER;
BEGIN
    FOR sub_id IN SELECT subject.id FROM subject LOOP
        WITH candidate AS (
            SELECT writtenwork.school_child_id, SUM(writtentask.grade) AS grade
            FROM writtentask
            JOIN writtenwork ON writtentask.written_work_id = writtenwork.id
            JOIN variant ON writtenwork.variant_id = variant.id
            WHERE variant.subject_id = sub_id
            GROUP BY writtenwork.school_child_id
            ORDER BY grade DESC
            LIMIT 1
        )
        SELECT grade, school_child_id INTO max_grade, max_school_child_id FROM candidate;

        IF max_grade > (SELECT average_grade FROM winner) THEN
            UPDATE winner
            SET id = max_school_child_id, average_grade = max_grade;
        END IF;
    END LOOP;
END $$;
select * from winner;
```

Data Output | Messages | Notifications

SQL

	id integer	average_grade integer
1	307489	68

c. По предмету "Физика" указать средний балл по каждому варианту

```sql
create temporary table physics_id as (
    select subject.id from subject where subject.name='Физика'
);
with marks as (
    select variant.id as var_id, writtenwork.id as work_id, sum(writtentask.grade) as sum_grade
    from writtentask
    join writtenwork
        on writtenwork.id=writtentask.written_work_id
    join variant
        on variant.id=writtenwork.variant_id
    where variant.subject_id=(select physics_id.id from physics_id)
    group by var_id, work_id
)
select marks.var_id, avg(sum_grade) as average
from marks
group by var_id
order by average desc;
```

Data Output | Messages | Notifications

	var_id integer	average numeric
1	743	22.7382256297918949
2	710	22.7016393442622951
3	735	22.6680327868852459
4	709	22.6677852348993289
5	721	22.6458773784355180
6	712	22.6259143155694880
7	736	22.6203501094091904
8	716	22.5986842105263158
9	745	22.5937149270482604
10	744	22.5646036916395223
11	733	22.5644670050761421
12	737	22.5540691192865106

Total rows: 50 | Query complete 00:00:03.141

d. Вывести средний балл по профильной математике для каждой школы Москвы

```sql
with m_id as (
    select subject.id from subject
    where subject.name='Математика. Профильный уровень'
),
moscow_id as (
    select city.id from city where city.name='Москва'
)
select marks.school_id, avg(marks.grade) as average from (
select schoolchild.id, schoolchild.school_id, sum(writtentask.grade) as grade from writtentask
    join writtenwork on writtentask.written_work_id=writtenwork.id
    join variant on writtenwork.variant_id=variant.id
    join schoolchild on schoolchild.id=writtenwork.school_child_id
    join address on schoolchild.address_id=address.id
    where variant.subject_id=(select m_id.id from m_id) and address.city_id=(select id from moscow_id)
    group by schoolchild.id
) as marks
group by marks.school_id
order by average desc;
```

Data Output | Messages | Notifications

	school_id integer	average numeric
1	6307	18.5000000000000000
2	32527	18.4285714285714286
3	30247	18.2500000000000000
4	33667	18.2500000000000000
5	26827	17.5714285714285714
6	19987	17.5000000000000000
7	5167	17.5000000000000000
8	7447	17.5000000000000000
9	14287	17.2222222222222222
10	29107	17.0833333333333333
11	23407	17.0000000000000000

Total rows: 35 | Query complete 00:00:04.106