ЗМІСТ

ВСТУП

База даних - це організована структура, призначена для зберігання інформації. З поняттям бази даних тісно пов'язане поняття системи управління базою даних. Це комплекс програмних засобів, призначених для створення структури нової бази, редагування вмісту і візуалізації інформації.

Інформаційна система - це сукупність засобів збору, зберігання, передачі, оброблення інформації в певній предметній області для досягнення поставленої мети у процесі управління.

Система управління базами даних (СУБД) - це комплекс програмних і мовних засобів, необхідних для створення баз даних, підтримання їх в актуальному стані та організації пошуку в них необхідної інформації.

Побудова будь-якої інформаційної системи починається зі створення предметної області, яка може бути реалізована у вигляді баз і сховищ даних. Тому для того, щоб деяку предметну область представити в базі даних, потрібно виділити істотні поняття, необхідні користувачу, а також зв’язки між ними.

Метою даної роботи є задоволення створення спеціалізованої СУБД, розрахованої на управління заздалегідь певною структурою інформації і рішення цілком певного і обмеженого кола завдань для рекламного агентства, яка одночасно дозволяла не відволікатися майбутнім користувачам на вивчення питань пов'язаних з базами даних і засобами управління ними.

Серед засобів визначення та маніпулювання даними існує мова SQL (Structured Query Language), перевага якої полягає в тому, що вона може використовуватися і як мова запитів, і як підмова даних, та дозволяє будувати як локальні, так і розподілені інформаційні системи .

В наш час бази даних є актуальними та широко використовуваними. За допомогою СУБД з’являється можливість відобразити необхідний віртуальний світ та вирішити необхідні проблеми.

1 ПОСТАНОВКА ЗАДАЧІ

Задачею даної курсової роботи є розробка проекту бази даних для зберігання та маніпулювання даними в закладі. База даних повинна бути спроектована з урахуванням реалізації запитів для отримання інформації, відповідно до завдань, які розв'язуються над предметною областю.

При виконані завдання була поставлена задача створення бази даних для підтримки діяльності автошколи, яка повинна надавати можливість:

- зберігати інформацію про студентів (анкетні дані, місце навчання/роботи), про викладачів (анкетні дані та категорії навчання), про навчання (обсяг занять, вартість, форма контролю, додаткові відомості);
- вести журнал фіксації результатів іспитів, списки за групами, результати внутрішньошкільного іспиту, створювати свідоцтво про закінчення автошколи;
- маніпулювати даними студентів, викладачів, груп та результатів навчання;

Створена база даних вирішує питання формування великої кількості інформації з якою можна зустрітися при створенні автоматизованих систем обліку студентів.

Використання цієї бази даних зробить працю з великою кількістю інформації набагато легше. База даних автоматизує багато процесів та робить роботу з даними значно легшою для адміністрації школи.

Функціонал та можливості розробленої бази, що були перелічені вище повинні також забезпечуватися певними програмними та функціональними можливостями, що надаються середовищем розробки.

Використання та комбінування даних засобів дозволить створити ефективну базу даних, що буде зручною у використанні, відносно простою та логічною у побудові, корисною для організації що її використовує.

Актуальність роботи полягає в можливості її практичного застосування у автошколах, що значно полегшить та прискорить работу з ведення обліку курсантів.

2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ

Перед проектуванням бази даних, потрібно розібрати предметну область і які сутності потрібно створити.

Опис сутностей:

1) Сутність «Преподаватель» - містить в собі інформацію про викладачів автошколи.

2) Сутність «Инструктор» - містить в собі інформацію про інструкторів автошколи, які приймають практичну частину іспиту у студентів.

3) Сутність «Курс_обучения» - містить в собі інформацію про всі доступні курси навчання автошколи.

4) Сутність «Группа» - містить в собі інформацію про всі групи студентів, які навчаються в автошколі.

5) Сутність «Журнал» - містить в собі всі результати іспитів студентів, як теоретичні так і практичні.

6) Сутність «Студент» - містить в собі інформацію про студента, що поступили в автошколу.

7) Сутність «Тип_экзамена» - містить в собі інформацію про види іспитів, доступні для здачі в автошколі

8) Сутність «Свидетельство_об_окончании» - містить в собі інформацію, яку повинен отримати студент після закінчення курсу та успішного складання іспиту.

9) Сутність «Анкета_студента» - містить в собі детальну інформацію про особу студента.

10) Сутність «Заявка_в_автошколу» - містить в собі інформацію про ще не поступили до автошколи студента.

Після створення сутностей, потрібно вирішити, яка кількість атрибутів буде існувати в кожній таблиці і який тип даних належить до кожного атрибута. Увесь результат представлено в табл. 2.1.

Таблиця 2.1 – Таблиці та їх атрібути

Ім'я сутності	Назва трибуту	Тип даних	Ключ
Преподаватель	id_преподавателя	int	Первинний
	Фамилия	char	
	Имя	char	
	Отчество	char	
	Наличие_частных занятий	char	
	Зарплата	decimal	
	Адресс_проживения	char	
	Высшее_образование	char	
Инструктор	id_инструктора	int	Первинний
	Фамилия	char	
	Имя	char	
	Отчество	char	
	Опыт_работы	int	
	Опыт_вождения	int	
	Категория_транспорта	char	
	Зарплата	decimal	
Курс_обучения	id_курса	int	Первинний
	Категория_курса	char	
	Стоимость	int	
	Продолжительность	int	
Студент	id_студента	int	Первинний
	Группа	int	Зовнішній
	Анкета	int	Зовнішній
	Количество_пропусков	int	
	Состояние_оплаты_за_курс	char	
Журнал	id_журнала	int	Первинний
	Студент	int	Зовнішній
	Экзамен	int	Зовнішній
	Дата_прохождения	date	
	Набранный_балл	int	
	Инструктор	int	Зовнішній
Заявка_в_автошколу	id_заявки	int	Первинний
	Анкета	int	Зовнішній
	Необходимая_категория	char	
Анкета_студента	id_анкеты	int	Первинний
	Фамилия	char	
	Имя	char	
	Отчество	char	

	Дата_рождения	date	
	Адресс_проживания	char	
	Место_учебы/работы	varchar	
Свидетельство_ об_окончании	id_свидетельства	int	Первинний
	Пройденный_курс	int	Зовнішній
	Студент	int	Зовнішній
	Результат_экзамена	int	Зовнішній
Тип_экзамена	id_экзамена	int	Первинний
	Категория_экзамена	char	
	Система_оценивания	char	
	Стоимость	decimal	
Группа	id_группы	int	Первинний
	Номер_группы	int	
	Количество_учащихся	int	
	Курс	int	Зовнішній
	Дата_начала	date	
	Расписание_занятий	char	
	Преподаватель	int	Зовнішній

Завершивши попередній процес, можна приступити до створення схеми-даних, у якій потрібно реалізувати зв’язки, такі як: один-до-одного, один-до-багатьох, багато-до-багатьох, між таблицями. Також можна позначити первинні і зовнішні ключі за допомогою яких і створюються зв’язки. Для даної БД представлена схема-даних (див. рис. 1.1) .

Зв’язок "один до одного" припускає, що в кожен момент часу кожному елементу (кортежу) А відповідає 0 або 1 елементів (кортежів) В. Наприклад, студент – анкета студента.

Зв’язок " багатьох до одного" полягає в тому, що в кожен момент часу кожному елементу (кортежу) А відповідає декілька елементів (кортежів) В. Наприклад, студент – группа.

Зв’язок "багато до багатьох" полягає в тому. що в кожен момент часу безлічі елементів А відповідає безліч елементів В. Цей тип зв’язку в реляційних БД безпосередньо не підтримується та нормалізація проводиться через третю таблицю. Наприклад, викладач автошколи – студент автошколи.

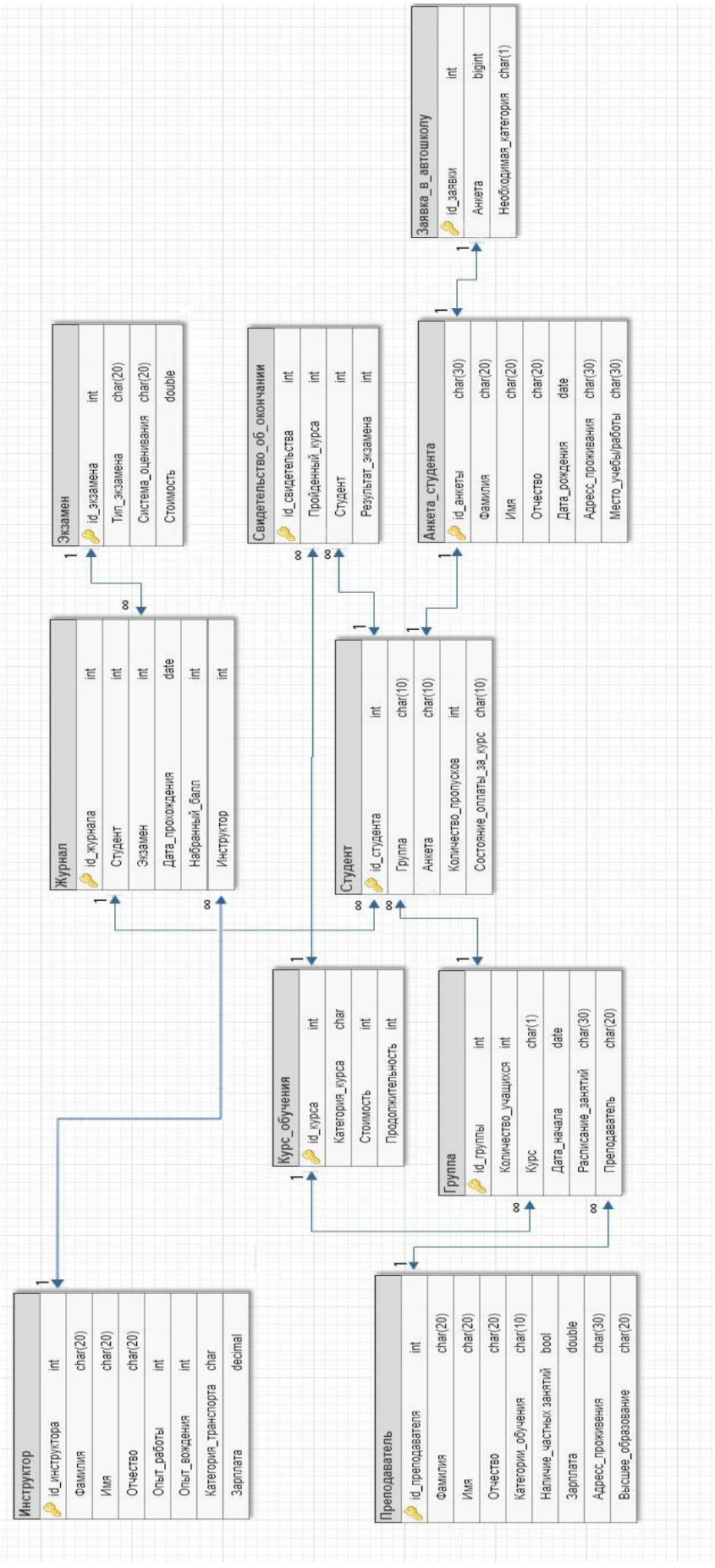

Рисунок 2.1 - Схема даних

3 ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

При виконанні наданого завдання була використана Система Управління Базами Даних PostgreSQL 9.6 під управлінням pgAdmin 4 в ОС Windows.

PostgreSQL – це об'єктно-реляційна система керування базами даних (СКБД). Вонаа надає безліч різних можливостей, досить надійна і має хороші характеристики по продуктивності. Вона працює практично на всіх UNIX-платформах, включаючи UNIX-подібні системи, такі як FreeBSD і Linux. Крім того, PostgreSQL вільно поширюється і має відкритий вихідний код.

Фундаментальна характеристика об'єктно-реляційної бази даних - це підтримка об'єктів і їх поведінки, включаючи типи даних, функції, операції, домени і індекси. Це робить Postgres неймовірно гнучким і надійним. Серед іншого, він вміє створювати, зберігати та видавати складні структури даних.

Існує великий список типів даних, які підтримує Postgres. Крім числових, з плаваючою точкою, текстових, булевих і інших очікуваних типів даних (а також безлічі їх варіацій), PostgreSQL може похвалитися підтримкою uuid, грошового, що перераховується, геометричного, бінарного типів, мережевих адрес, бітових рядків, текстового пошуку, xml, json, масивів, композитних типів і діапазонів, а також деяких внутрішніх типів для ідентифікації об'єктів і розташування логів.

pgAdmin - це програмне забезпечення з відкритим кодом для розробки та адміністрування баз даних PostgreSQL та похідних баз даних, таких як EnterpriseDB Postgres Plus Advanced Server або Greenplum. Графічний інтерфейс користувача полегшує адміністрування баз даних. Редактор SQL Query містить графічний EXPLAIN, який дозволяє створювати більш потужні запити. Власне з'єднання з PostgreSQL дозволяє графічному інтерфейсу отримати доступ до всіх функцій PostgreSQL.

Таким чином, вважаючи усі ці переваги, було обрано PostgreSQL для написання бази данних.

4 СТВОРЕННЯ БАЗИ ДАНИХ

4.1 Створення таблиць

Для визначення таблиць МВД *SQL* має такі команди:

CREATE TABLE— для створення таблиці,

ALTER TABLE— для модифікації таблиці і DROP TABLE— для її знищення.

Найбільш простий варіант команди створення таблиці:

CREATE TABLE *таблиця*

(*поле тип_поля* [DEFAULT *значення*][, *поле тип_поля* [DEFAULT *значення*][,…]]);

Операнди *таблиця* і *поле* визначають імена відповідних елементів. Як *тип_поля* вказу-ється або ім’я домену, або один з базових типів. Причому будь-який числовий тип може вка-зуватися без розміру, який буде прийнятий за замовчуванням, а параметр DEFAULT *скасовує аналогічне значення* домену.

Обмежень поля може бути кілька, записаних для поля через пробіл:

NOT NULL – не пусте

NULL – пусті значения дозволені

UNIQUE - значеня поля унікальні

PRIMARY KEY - первинний ключ

CHECK - умова на значення

REFERENCES пов’язана_таблиця [(пов’язане _поле)] [ON DELETE action] [ON UPDATE action] - ВИЗНАЧЕННЯ зв'язку між таблицями через зовнішній ключ action (NO ACTION; RESTRICT; CASCADE; SET NULL; SET DEFAULT)

Обмежень таблиці може бути декілька, записаних для таблиці через кому:

UNIQUE (имя_поля [, ...]) – уникальне значення PRIMARY KEY (имя_поля [, ...]) – первинний ключ CHECK (expression) обеження на значення

FOREIGN KEY (имя_поля [, ...]) REFERENCES зв’язана_таблиця [(зв’язане_поле [, ...])] [ON DELETE action] [ON UPDATE action] – визначення зв’язку між таблицями

Таблиці формуються в бази даних, а колекція баз даних, керована однією копією сервера PostgreSQL називається кластером баз даних. Команда створення таблиці: усі приклади в Додатку А.

Приклад: Приклад створення таблиці студентів:

```
CREATE TABLE "Студент" (
        "id_студента" INT DEFAULT NEXTVAL ('s_Student') UNIQUE,
        "Группа" int NOT NULL,
        "Анкета" int NOT NULL UNIQUE,
        "Количество_пропусков" int NOT NULL,
        "Состояние_оплаты_за_курс" char(30) NOT NULL,
        CONSTRAINT Студент_pk PRIMARY KEY ("id_студента")
);
```

4.2 Створення представлень

Представлення (VIEW) — об’єкт, що не містить власних даних. Це іменована похід на віртуальна таблиця, що не може існувати сама по собі, а визначається в термінах однієї або декількох іменованих таблиць (базових таблиць або інших представлень).

У загальному випадку термін похідна таблиця позначає таблицю, що визначається в термінах інших таблиць і, в остаточному підсумку, у термінах базових таблиць, тобто є результатом виконання яких-небудь реляційних виразів над ними. Базова таблиця — це така таблиця, що не є похідною.
Більш того, будь-які зміни в основній таблиці будуть автоматично і негайно видані через таке „вікно“. І навпаки, зміни в представленні будуть автоматично і негайно застосовані до його базової таблиці.

У дійсності ж представлення — це запити, які виконуються щоразу, коли представлення є об’єктом команди *SQL*.

1) Створення представлення для перегляду важливою інформації про курсантів в группах

```
CREATE OR REPLACE VIEW public.full_info AS
SELECT  ans."Имя", ans."Фамилия", ans."Отчество",
            gr."Номер_группы", gr."Дата_начала",
            curs."Категория_курса", st."Количество_пропусков"
FROM "Студент" st, "Анкета_студента" ans, "Курс_обучения" curs, "Группа" gr
WHERE st."Анкета" = ans."id_анкеты" AND st."Группа" = gr."id_группы" AND
            gr."Курс" = curs."id_курса"
ORDER BY gr."Номер_группы";
```

	Имя character(20)	Фамилия character(20)	Отчество character(20)	Номер_группы integer	Дата_начала date	Категория_курса character(1)	Количество_пропусков integer
1	Лев	Многогрешный	Дмитриевич	111	2018-11-25	B	1
2	Аркадий	Баранов	Юхимович	111	2018-11-25	B	5
3	Зуфар	Громов	Сергеевич	111	2018-11-25	B	2
4	Иммануил	Павлик	Станиславович	111	2018-11-25	B	0
5	Мирослав	Прохоров	Грегориевич	111	2018-11-25	B	0
6	Юлий	Гелетей	Вадимович	111	2018-11-25	B	3
7	Никита	Чикольба	Виталиевич	112	2017-12-01	A	5
8	Марк	Дубченко	Грегориевич	112	2017-12-01	A	3
9	Милан	Пархоменко	Максимович	112	2017-12-01	A	0
10	Добрыня	Архипов	Львович	113	2018-12-12	B	0
11	Клим	Суворов	Львович	113	2018-12-12	B	1
12	Аким	Дорофеев	Ярославович	113	2018-12-12	B	7
13	Ростислав	Новиков	Данилович	113	2018-12-12	B	2
14	Орест	Белов	Иванович	113	2018-12-12	B	8
15	Стефан	Новиков	Вадимович	114	2016-10-25	B	1
16	Леонард	Зыков	Викторович	114	2016-10-25	B	4
17	Остин	Фролов	Владимирович	114	2016-10-25	B	6
18	Заур	Несвитайло	Иванович	114	2016-10-25	B	2
19	Доминик	Гелетей	Платонович	114	2016-10-25	B	10
20	Никита	Марков	Виталиевич	115	2018-11-22	C	7
21	Жигер	Фёдоров	Петрович	115	2018-11-22	C	3
22	Дан	Виноградов	Сергеевич	115	2018-11-22	C	1
23	Игнатий	Матвеев	Артёмович	115	2018-11-22	C	0
24	Емельян	Кириленко	Грегориевич	115	2018-11-22	C	4
25	Добрыня	Архипов	Львович	115	2018-11-22	C	7
26	Гавриил	Исаков	Евгеньевич	115	2018-11-22	C	3

Рисунок 4.1 – Представлення «full_info»

2)Створення представлення для перегляду результатів теоретичної частини экзамену студентів в порядку від кращого к гіршому.

```
CREATE OR REPLACE VIEW public.top_theor_exam_log AS
    SELECT  ans."Имя", ans."Фамилия", ans."Отчество",
            gr."Номер_группы", exam."Категория_экзамена",
            log."Дата_прохождения",    log."Набранный_балл"
    FROM "Студент" st, "Анкета_студента" ans, "Журнал" log, "Тип_экзамена" exam,
            "Группа" gr
    WHERE st."Анкета" = ans."id_анкеты" AND st."Группа" = gr."id_группы"
    AND log."Экзамен" =  exam."id_экзамена"  AND log."Студент" =  st."id_студента"
            AND exam."Категория_экзамена" LIKE 'Теория%'
    ORDER BY log."Набранный_балл"  DESC;
```

	Имя character(20)	Фамилия character(20)	Отчество character(20)	Номер_группы integer	Категория_экзамена character(20)	Дата_прохождения date	Набранный_балл integer
1	Никита	Чикольба	Виталиевич	112	Теория категории A	2018-03-02	97
2	Клим	Суворов	Львович	113	Теория категории A	2018-03-07	80
3	Остин	Фролов	Владимирович	114	Теория категории B	2017-01-26	76
4	Стефан	Новиков	Вадимович	114	Теория категории B	2017-01-28	73
5	Заур	Несвитайло	Иванович	114	Теория категории B	2017-01-29	69
6	Заур	Несвитайло	Иванович	114	Теория категории B	2017-01-29	69
7	Милан	Пархоменко	Максимович	112	Теория категории A	2018-03-03	65
8	Доминик	Гелетей	Платонович	114	Теория категории B	2017-01-26	53
9	Марк	Дубченко	Грегориевич	112	Теория категории A	2018-03-01	50
10	Леонард	Зыков	Викторович	114	Теория категории B	2017-01-28	33

Рисунок 4.2 – Представлення «top_theor_exam_log»

3)Створення представлення для перегляду результатів практичної частини экзамену студентів в порядку від кращого к гіршому.

```
CREATE OR REPLACE VIEW public.top_prakt_exam_log AS
    SELECT  ans."Имя", ans."Фамилия", ans."Отчество",
            gr."Номер_группы", exam."Категория_экзамена",
            log."Дата_прохождения",    log."Набранный_балл"
    FROM "Студент" st, "Анкета_студента" ans, "Журнал" log, "Тип_экзамена" exam,
            "Группа" gr
    WHERE st."Анкета" = ans."id_анкеты" AND st."Группа" = gr."id_группы"
    AND log."Экзамен" =  exam."id_экзамена"  AND log."Студент" =  st."id_студента"
                AND exam."Категория_экзамена" LIKE 'Практика%'
    ORDER BY log."Набранный_балл"  DESC;
```

	Имя character(20)	Фамилия character(20)	Отчество character(20)	Номер_группы integer	Категория_экзамена character(20)	Дата_прохождения date	Набранный_балл integer
1	Заур	Несвитайло	Иванович	114	Практика категории B	2017-01-28	29
2	Милан	Пархоменко	Максимович	112	Практика категории A	2018-03-03	29
3	Доминик	Гелетей	Платонович	114	Практика категории B	2017-01-25	26
4	Никита	Чикольба	Виталиевич	112	Практика категории A	2018-03-01	25
5	Стефан	Новиков	Вадимович	114	Практика категории B	2017-01-27	23
6	Остин	Фролов	Владимирович	114	Практика категории B	2017-01-25	19
7	Марк	Дубченко	Грегориевич	112	Практика категории A	2018-03-02	12
8	Клим	Суворов	Львович	113	Практика категории A	2018-03-04	10
9	Леонард	Зыков	Викторович	114	Практика категории B	2017-01-27	5

Рисунок 4.3 – Представлення «top_prakt_exam_log»

4)Створення представлення для перегляду випускників автошколи.

```
CREATE OR REPLACE VIEW public.certificates AS
    SELECT  ans."Имя", ans."Фамилия", curs."Категория_курса", gr."Дата_начала",
            log."Дата_прохождения",    log."Набранный_балл"
    FROM "Студент" st, "Анкета_студента" ans, "Журнал" log, "Курс_обучения" curs,
            "Группа" gr, "Свидетельство_об_окончании" cert, "Тип_экзамена" exam
    WHERE cert."Студент" = st."id_студента" AND st."Анкета" = ans."id_анкеты" AND
            st."Группа" = gr."id_группы" AND log."Студент" = cert."Результат_экзамена"
            AND cert."Пройденный_курс" = curs."id_курса"
            AND log."Экзамен" =  exam."id_экзамена" AND exam."Категория_экзамена"
            LIKE 'Теория%'
            AND log."Набранный_балл" > 60
    ORDER BY log."Дата_прохождения";
```

	Имя character(20)	Фамилия character(20)	Категория_курса character(1)	Дата_начала date	Дата_прохождения date	Набранный_балл integer
1	Доминик	Гелетей	B	2016-10-25	2018-03-03	65
2	Клим	Суворов	A	2018-12-12	2018-03-02	97
3	Остин	Фролов	B	2016-10-25	2017-01-28	73
4	Заур	Несвитайло	B	2016-10-25	2017-01-26	76

Рисунок 4.4 – Представлення «certificates»

4.3.Створення тригерів

Тригери мають певну подібність (тобто аналогічні) збереженим процедурам, але тим же часом відрізняються від них.

Тригери — це підпрограми, написані SQL, які виконуються тоді, коли відбувається певна подія, спрямована до конкретної таблиці.

Такою подією може бути виконання якоїсь команди відновлення ММД SQL: вставки, відновлення або видалення даних. Відповідно кожен тригер асоціюється з конкретною операцією і конкретною таблицею. Така відповідність встановлюється при створенні три-гера командою CREATE TRIGGER. Формат цієї команди в різних СУБД трохи відрізняється. Тому спочатку розглянемо побудову тригерів в стандартному SQL, а потім — особливості роботи з тригерами з використанням діалекту PostgreSQL.

Тригери можуть використовуватися в наступних областях функціонування додатків:

— реалізація контролю за всіма діями користувачів БД;

— забезпечення цілісності змісту БД;

— реалізація складних правил роботи програм;

— забезпечення складних правил безпеки даних;

— автоматичне створення значень у полях таблиць БД.

Виконання процедури (тригера) обумовлено дією по модифікації даних: додаванням INSERT, видаленням DELETE рядка в заданій таблиці, або зміною UPDATE даних в певному стовпці заданої таблиці реляційної бази даних. Тригери застосовуються для забезпечення цілісності даних і реалізації складної бізнес-логіки. Тригер запускається сервером автоматично при спробі зміни даних в таблиці, з якою він пов'язаний. Всі вироблені їм модифікації даних розглядаються як виконуються в транзакції, в якій виконано дію, яка викликала спрацьовування тригера.

Відповідно, в разі виявлення помилки або порушення цілісності даних може статися відкат цієї транзакції. Момент запуску тригера визначається за допомогою ключових слів BEFORE (тригер запускається до виконання пов'язаного з ним події, наприклад, до додавання запису) або AFTER (після події). У разі, якщо тригер викликається до події, він може внести зміни в модифікуються подією запис (звичайно, за умови, що подія - не вилучення запису). Деякі СУБД накладають обмеження на оператори, які можуть бути використані в тригері (наприклад, може бути заборонено вносити зміни в таблицю, на якій «висить» тригер, і т. П.)

1) Тригер видаляє Анкету студента і заявку в автошколу при видаленні студента і зменшує кількість студентів в групі на 1:

```
CREATE OR REPLACE FUNCTION DeleteStudent() RETURNS TRIGGER
        AS $$       BEGIN
DELETE FROM "Заявка_в_автошколу" ank WHERE ank."id_заявки" = old."id_студента";
DELETE FROM "Анкета_студента" ank WHERE ank."id_анкеты" = old."id_студента";
UPDATE "Группа" SET "Количество_учащихся" = "Количество_учащихся" - 1 WHERE
"id_группы" = old."Группа";
ALTER  SEQUENCE s_Instructor INCREMENT BY -1;
        RETURN old;
        END; $$
LANGUAGE 'plpgsql';
CREATE TRIGGER DelStud
        AFTER DELETE ON "Студент");
        FOR EACH ROW EXECUTE PROCEDURE DeleteStudent();
```

2) Тригер судячи з доданою заявці в автошколу додає студента в групу з потрібною категорією і найменьше кількістю студентів (якщо потрібної категорії ні - не додає) і збільшує кількість людей в групі на 1:

```
--without var
CREATE OR REPLACE FUNCTION AddStudent()        RETURNS TRIGGER
        AS $$
        BEGIN
INSERT INTO "Студент" VALUES (NEXTVAL ( 's_Student' ), (SELECT "id_группы" FROM "Группа"
        WHERE "Количество_учащихся" = (SELECT MIN("Количество_учащихся") FROM "Группа"
            WHERE "Курс" = (SELECT "id_курса" FROM "Курс_обучения"
                WHERE "Категория_курса" = new."Необходимая_категория"))), new."Анкета", 0, 'Не уплачено');

UPDATE "Группа" SET "Количество_учащихся" = "Количество_учащихся" + 1
        WHERE "id_группы" = (SELECT "id_группы" FROM "Группа"
            WHERE "Количество_учащихся" = (SELECT MIN("Количество_учащихся") FROM "Группа" WHERE "Курс" = (SELECT "id_курса" FROM "Курс_обучения"
                WHERE "Категория_курса" = new."Необходимая_категория")));
        RETURN new;
        END; $$
LANGUAGE 'plpgsql';

CREATE TRIGGER AddStud
        AFTER INSERT ON "Заявка_в_автошколу"
        FOR EACH ROW EXECUTE PROCEDURE AddStudent();
--with var
CREATE OR REPLACE FUNCTION AddStudent()        RETURNS TRIGGER
        AS $$
        DECLARE
            id_group int;
        BEGIN
id_group = (SELECT "id_группы" FROM "Группа"
        WHERE "Количество_учащихся" = (SELECT MIN("Количество_учащихся") FROM "Группа"
            WHERE "Курс" = (SELECT "id_курса" FROM "Курс_обучения"
                WHERE "Категория_курса" = new."Необходимая_категория")));

INSERT INTO "Студент" VALUES (NEXTVAL ( 's_Student' ), id_group, new."Анкета", 0, 'Не уплачено');
UPDATE "Группа" SET "Количество_учащихся" = "Количество_учащихся" + 1 WHERE "id_группы" = id_group;
        RETURN new;
        END; $$
LANGUAGE 'plpgsql';

CREATE TRIGGER AddStud
        AFTER INSERT ON "Заявка_в_автошколу"
        FOR EACH ROW EXECUTE PROCEDURE AddStudent();
```

3) Тригер судячи з доданого картежу в журнал, при потрібному кількості балів автоматично формує свідоцтво про закінчення:

```
CREATE OR REPLACE FUNCTION AddCertificate() RETURNS TRIGGER
	AS $$	BEGIN
IF EXISTS ((SELECT "Категория_экзамена" FROM "Тип_экзамена" WHERE "id_экзамена" = new."Экзамен" AND "Категория_экзамена" LIKE 'Теория%' AND new."Набранный_балл" > 60)) THEN
	INSERT INTO "Свидетельство_об_окончании" VALUES (NEXTVAL ( 's_Certificate' ), (SELECT "id_курса" FROM "Курс_обучения"
		WHERE "id_курса" = (SELECT "Курс" FROM "Группа"
			WHERE "id_группы" = (SELECT "Группа" FROM "Студент" WHERE "id_студента" = new."Студент"))), new."Студент", new."id_журнала" );
			return new;
			END IF;

IF EXISTS ((SELECT "Категория_экзамена" FROM "Тип_экзамена" WHERE "id_экзамена" = new."Экзамен" AND "Категория_экзамена" LIKE 'Практика%' AND new."Набранный_балл" > 30)) THEN
		INSERT INTO "Свидетельство_об_окончании" VALUES (NEXTVAL ( 's_Certificate' ), (SELECT "id_курса" FROM "Курс_обучения"
		WHERE "id_курса" = (SELECT "Курс" FROM "Группа"
			WHERE "id_группы" = (SELECT "Группа" FROM "Студент" WHERE "id_студента" = new."Студент"))), new."Студент", new."id_журнала" );
			return new;
			END IF;

			RAISE EXCEPTION 'Студент не сдал экзамен :(';
			return new;
	END; $$
LANGUAGE 'plpgsql';

CREATE TRIGGER AddCert
	AFTER INSERT ON "Журнал"
	FOR EACH ROW EXECUTE PROCEDURE AddCertificate();
```

4) Трігер перевіряє на правильність введених в журнал оцінок (максимум 100 і не більше 50):

```
	CREATE OR REPLACE FUNCTION CheckMark()
	RETURNS TRIGGER
	  AS $$	BEGIN
IF EXISTS ((SELECT "Категория_экзамена" FROM "Тип_экзамена" WHERE "id_экзамена" = new."Экзамен" AND "Категория_экзамена" LIKE 'Теория%' AND new."Набранный_балл" > 100)) THEN
			RAISE EXCEPTION 'Ошибка! Балл за теорию не может быть больше > 100.';
			return old;
			END IF;
```

```
IF EXISTS ((SELECT "Категория_экзамена" FROM "Тип_экзамена" WHERE "id_экзамена" = new."Экзамен" AND "Категория_экзамена" LIKE 'Практика%' AND new."Набранный_балл" > 50)) THEN
                    RAISE EXCEPTION 'Ошибка! Балл за практику не может быть больше > 50.';
                    return old;
                    END IF;
                    RETURN new;
        END; $$
LANGUAGE 'plpgsql';

CREATE TRIGGER CheckM
        BEFORE INSERT ON "Журнал"
        FOR EACH ROW EXECUTE PROCEDURE CheckMark();
```

5)Триггер проверяет, чтобы категория курса была введена правильно:

```
        CREATE OR REPLACE FUNCTION CheckCurs()
        RETURNS TRIGGER
                AS $$ BEGIN
        IF EXISTS (SELECT old."Категория_курса" FROM "Курс_обучения" WHERE NOT old."Категория_курса" = 'A'
                OR NOT old."Категория_курса" = 'B'
                OR NOT old."Категория_курса" = 'C'
                OR NOT old."Категория_курса" = 'D') THEN
                                RAISE EXCEPTION 'Ошибка! Неправильно введена категория курса';
                                return old;
                                END IF;
                return new;
                END; $$
        LANGUAGE 'plpgsql';

CREATE TRIGGER CheckC
        BEFORE INSERT ON "Курс_обучения"
        FOR EACH ROW EXECUTE PROCEDURE CheckCurs ();
```

4.4.Створення функцій

На відміну від збережених процедур стандартного SQL, наведених вище, та, навіть, на відміну від СУБД Oracle, яка містить два типи підпрограм — процедури (аналогічні стандар-тним) і функції, PLpg/SQL підтримує тільки користувальницькі функції.

На відміну від збережених процедур стандартного SQL, наведених вище, та, навіть, на відміну від СУБД Oracle, яка містить два типи підпрограм — процедури (аналогічні стандар-тним) і функції, PLpg/SQL підтримує тільки користувальницькі функції.

Функції виконуються на сервері, а не на клієнті БД. Хоча вони можуть бути написані на чистому SQL, реалізація додаткової логіки, наприклад, умовних переходів і циклів, виходить за рамки власне SQL і вимагає використання деяких мовних розширень. Функції можуть писатися з використанням однієї з наступних мов:

1. вбудованої процедурної мови PL/pgSQL, яка багато в чому аналогічна мові PL/SQL, що використовується в СУБД Oracle;
2. скриптової мови - PL/Lua, PL/LOLCODE, PL/Perl, plPHP, PL/Python, PL/Ruby, PL/sh, PL/Tcl і PL/Scheme;
3. класичної мови - C, C++, Java (через модуль PL/Java);
4. статистичної мови R (через модуль PL/R).

PostgreSQL допускає використання функцій, які повертають набір записів, який можна використовувати так само, як і результат виконання звичайного запиту.

Функції можуть виконуватися як з правами їх творця, так і з правами поточного користувача.

Іноді функції ототожнюються з збереженими процедурами, однак між цими поняттями є різниця.

Набір функцій, які були реалізовані впродовж виконання курсової роботи:

1) Функція, яка змінює в певного абітурієнта оцінку за певний іспит:

```
CREATE FUNCTION changeExamMark(int, int)
RETURNS varchar
AS $$
BEGIN
UPDATE "Журнал" SET "Набранный_балл"=$2 where "id_журнала"=$1;
return (SELECT "Фамилия" FROM "Анкета_студента" WHERE "id_анкеты" =
(SELECT "Анкета" FROM "Студент" WHERE "id_студента"= (SELECT "Студент"
FROM "Журнал" WHERE "Студент" = $1)));
END;
$$
LANGUAGE 'plpgsql';
```

2) Функція, яка шукає та змінює викладача у певній групі на вікладача з такою ж категорією навчання:

```
CREATE OR REPLACE FUNCTION findProfessor(int)
RETURNS int
AS $$
BEGIN
UPDATE "Группа" SET "Преподаватель" = (SELECT "id_преподавателя" FROM
"Преподаватель" WHERE "Категории_обучения" = (SELECT "Категория_курса"
FROM "Курс_обучения" WHERE "id_курса" = (SELECT "Курс" FROM "Группа"
WHERE "id_группы" = $1))) WHERE "id_группы" = $1;
return (SELECT "id_преподавателя" FROM "Преподаватель" WHERE
"Категории_обучения" = (SELECT "Категория_курса" FROM "Курс_обучения"
WHERE "id_курса" = (SELECT "Курс" FROM "Группа" WHERE "id_группы" = $1)));
END; $$
LANGUAGE 'plpgsql';
```

3) Функція, которая подсчитывает стоимость всего курса, с учетом цены за единоразую сдачу экзаменов

```
CREATE OR REPLACE FUNCTION totalPrice(catagory char(30))
RETURNS int
AS $$
DECLARE
        total_sum int;
BEGIN
 total_sum = (SELECT "Стоимость" FROM "Курс_обучения" WHERE
"Категория_курса" = 'A') +
                (SELECT "Стоимость" FROM "Тип_экзамена" WHERE
"Категория_экзамена" LIKE '%' || catagory || '%' LIMIT 1);
return total_sum;
END; $$
LANGUAGE 'plpgsql';
```

5 МАНІПУЛЮВАННЯ ДАНИМИ

Хоча спочатку мова SQL була створена саме для запитів на вибірку даних (структурована мова запитів), але її призначення значно розширилось, і зараз за стандартом мови SQL вона використовується в якості DML (мови маніпулювання даними), тобто має команди для додавання записів у таблиці, оновлення даних в таблицях (зміни значень), та усунення (видалення) записів з таблиці.

Мова маніпулювання даними - командна мова, що забезпечує виконання основних операцій по роботі з даними: введення, модифікацію і вибірку даних за запитами. До базових ср-вам маніпулювання даними мови SQL відносяться "пошукові" варіанти операторів UPDATE і DELETE. Ці варіанти називаються пошуковими, тому що при завданні відповідної операції задається логічне умова, що накладається на рядки адресується оператором таблиці, які д.б.н. піддані модифікації або видалення. Крім того, в таку категорію мовних засобів входить оператор INSERT, що дозволяє додавати рядки в існуючі таблиці.

Для видалення записів використовується команда DELETE мови SQL.
DELETE FROM [*DatabaseName!*]*TableName* [WHERE *FilterCondition1* [AND | OR *FilterCondition2* ...]] FROM [*DatabaseName!*]*TableName]*

Вказує таблицю, в якій записи відмічаються на видалення. Тут *TableName* – назва (ім'я) таблиці, *DatabaseName* – ім'я бази даних (вказується, якщо таблиця не в активній базі даних).

WHERE *FilterCondition*

Вказує, що тільки визначені умовою *FilterCondition* записи відмічаються на видалення. *FilterCondition* – звичайний предикат, якій може включати вкладені запити, але не більше двох запитів.Як і команда DELETE мови FoxPro, команда DELETE - SQL тільки відмічає записи на усунення. Для фізичного видалення відміченних записів потрібно використати команду PACK.

Команда UPDATE – SQL оновлює записи в таблиці новими значеннями.

UPDATE

[*DatabaseName1!*]*TableName1* SET *Column_Name1 = eExpression1* [, *Column_ Name2 = eExpression2* ...] [WHERE *FilterCondition*] FROM [*DatabaseName!*]*TableName]*

Вказує таблицю, в якій оновлюються записи. Тут *TableName* – назва (ім'я) таблиці, *DatabaseName* – ім'я бази даних (вказується, якщо таблиця не в активній базі даних).

SET *Column_Name1 = eExpression1* [, *Column_Name2 = eExpression2*...]

Вказує атрибути (*Column_Name1*, *Column_Name2*), значення яких оновлюється новими значеннями (*eExpression1*, *eExpression2*). WHERE *FilterCondition*

Вказує, що тільки в визначених умовою *FilterCondition* записах оновлюються значення атрибутів. *FilterCondition* – звичайний предикат, якій може включати вкладені запити, але не більше двох запитів.

Вкладені запити в командах UPDATE – SQL та DELETE – SQL можна використовувати для визначення рядків таблиці, що видаляються або оновлюються, з використанням даних з інших таблиць.

Оператор SELECT є фактично найважливішим для користувача і найскладнішим оператором SQL. Він призначений для вибірки даних з таблиць, тобто він, власне, і реалізує одне з осн-х призначення БД - надавати інформацію користей-лю.

Структура команди Select ...

- from ... - таблиця, з якої будуть вилучатись дані;
- where ... - «горизонтальний» фільтр, умова на рядки;
- order by ... - критерій впорядкування рядків результатірующего таблиці
- group by ... - критерій групування рядків таблиці: рядки таблиці розбиваються на групи з однаковим значенням критерію, і кожна група дає єдиний рядок в вихідну таблицю;
- having ... - критерій фільтрації груп;
- into ... - куди і в якому вигляді записати результат.

В ході виконання курсової роботи було реалізовано 10 запитів:

1) Запит, який виводить всю інформацію про абітурієнтів:

```
SELECT ans."Имя",
   ans."Фамилия",
   ans."Отчество",
   gr."Номер_группы",
   exam."Категория_экзамена",
   log."Дата_прохождения",
   log."Набранный_балл"
  FROM "Студент" st,
   "Анкета_студента" ans,
   "Журнал" log,
   "Тип_экзамена" exam,
   "Группа" gr
  WHERE "Анкета" = ANY (SELECT "id_анкеты" FROM "Студент" WHERE "id_анкеты" = "Анкета")
                AND st."Группа" = gr."id_группы"
                    AND log."Экзамен" = exam."id_экзамена"
                        AND log."Студент" = st."id_студента"
  ORDER BY log."Набранный_балл" DESC;
```

	Имя character(20)	Фамилия character(20)	Отчество character(20)	Номер_группы integer	Категория_экзамена character(20)	Дата_прохождения date	Набранный_балл integer
1	Никита	Чикольба	Виталиевич	112	Теория категории A	2018-03-02	97
2	Клим	Суворов	Львович	113	Теория категории A	2018-03-07	80
3	Остин	Фролов	Владимирович	114	Теория категории B	2017-01-26	76
4	Стефан	Новиков	Вадимович	114	Теория категории B	2017-01-28	73
5	Заур	Несвитайло	Иванович	114	Теория категории B	2017-01-29	69
6	Заур	Несвитайло	Иванович	114	Теория категории B	2017-01-29	69
7	Милан	Пархоменко	Максимович	112	Теория категории A	2018-03-03	65
8	Доминик	Гелетей	Платонович	114	Теория категории B	2017-01-26	53
9	Марк	Дубченко	Грегориевич	112	Теория категории A	2018-03-01	50
10	Леонард	Быков	Викторович	114	Теория категории B	2017-01-28	33
11	Никита	Чикольба	Виталиевич	112	Практика категории A	2018-03-01	29
12	Милан	Пархоменко	Максимович	112	Практика категории A	2018-03-03	29
13	Заур	Несвитайло	Иванович	114	Практика категории B	2017-01-28	29
14	Доминик	Гелетей	Платонович	114	Практика категории B	2017-01-25	26
15	Стефан	Новиков	Вадимович	114	Практика категории B	2017-01-27	23
16	Остин	Фролов	Владимирович	114	Практика категории B	2017-01-25	19
17	Марк	Дубченко	Грегориевич	112	Практика категории A	2018-03-02	12
18	Клим	Суворов	Львович	113	Практика категории A	2018-03-04	10
19	Леонард	Быков	Викторович	114	Практика категории B	2017-01-27	5

Рисунок 5.1.- Результат першого запиту

2) Запит повертає середню кількість пропусків серед студентів у групах:

```
SELECT gr."Номер_группы",
   gr."Количество_учащихся",
   curs."Категория_курса",
      AVG(st."Количество_пропусков") AS AVG_abstents
FROM "Анкета_студента" ans,
   "Группа" gr,
   "Курс_обучения" curs,
    "Студент" st
WHERE gr."Курс" = curs."id_курса" AND st."Группа" = gr."id_группы"
GROUP BY gr."Номер_группы", gr."Количество_учащихся",
curs."Категория_курса";
```

	Номер_группы integer	Количество_учащихся integer	Категория_курса character(1)	avg_abstents numeric
1	111	6	B	1.8333333333333333
2	112	3	A	2.6666666666666667
3	113	4	B	3.6000000000000000
4	114	5	B	4.6000000000000000
5	115	7	C	3.5714285714285714

Рисунок 5.2.- Результат другого запиту

3) Запит виводить середню оцінку за теоретичну частину іспиту в кожній групі:

```
SELECT gr."Номер_группы",
   gr."Количество_учащихся",
   curs."Категория_курса",
      AVG(jer."Набранный_балл") AS AVG_mark
FROM "Анкета_студента" ans,
   "Группа" gr,
   "Курс_обучения" curs,
    "Студент" st,
    "Журнал" jer,
    "Тип_экзамена" exam
WHERE gr."Курс" = curs."id_курса" AND st."Группа" = gr."id_группы"
   AND jer."Студент" = st."id_студента"
            AND exam."id_экзамена" = jer."Экзамен"
                  AND exam."Категория_экзамена" LIKE 'Теория%'
GROUP BY gr."Номер_группы", gr."Количество_учащихся",
curs."Категория_курса";
```

	Номер_группы integer	Количество_учащихся integer	Категория_курса character(1)	avg_mark numeric
1	112	3	A	70.666666666666667
2	113	4	B	80.0000000000000000
3	114	5	B	62.166666666666667

Рисунок 5.3.- Результат третього запиту

4) Запит виводить середню оцінку за практичну частину іспиту в кожній групі:

```
SELECT gr."Номер_группы",
   gr."Количество_учащихся",
   curs."Категория_курса",
      AVG(jer."Набранный_балл") AS AVG_mark
FROM "Анкета_студента" ans,
   "Группа" gr,
   "Курс_обучения" curs,
    "Студент" st,
    "Журнал" jer,
    "Тип_экзамена" exam
WHERE gr."Курс" = curs."id_курса" AND st."Группа" = gr."id_группы"
   AND jer."Студент" = st."id_студента"
```

```
                AND exam."id_экзамена" = jer."Экзамен"
                        AND exam."Категория_экзамена" LIKE 'Практика%'
GROUP BY gr."Номер_группы", gr."Количество_учащихся",
curs."Категория_курса";
```

	Номер_группы integer	Количество_учащихся integer	Категория_курса character(1)	avg_mark numeric
1	112	3	A	23.3333333333333333
2	113	4	B	10.0000000000000000
3	114	5	B	20.4000000000000000

Рисунок 5.4.- Результат четвертого запиту

5) Запит виводить студентів, чий бал за теоретичну частину іспиту більше середньої серед всіх студентів:

```
SELECT ans."Имя",
   ans."Фамилия",
   ans."Отчество",
   gr."Номер_группы",
   exam."Категория_экзамена",
   jer."Набранный_балл"
FROM "Анкета_студента" ans,
   "Группа" gr,
    "Студент" st,
    "Журнал" jer,
    "Тип_экзамена" exam
WHERE jer."Студент" = st."id_студента"
  AND st."Группа" = gr."id_группы"
          AND exam."id_экзамена" = jer."Экзамен"
                  AND ans."id_анкеты" = st."Анкета"
                          AND exam."Категория_экзамена" LIKE 'Теория%'
   GROUP BY gr."Номер_группы", ans."Имя", ans."Фамилия", ans."Отчество",
exam."Категория_экзамена",  jer."Набранный_балл"
   HAVING jer."Набранный_балл" > (SELECT AVG("Набранный_балл") FROM
"Журнал");
```

	Имя character(20)	Фамилия character(20)	Отчество character(20)	Номер_группы integer	Категория_экзамена character(20)	Набранный_балл integer
1	Марк	Дубченко	Грегориевич	112	Теория категории A	50
2	Милан	Пархоменко	Максимович	112	Теория категории A	65
3	Никита	Чикольба	Виталиевич	112	Теория категории A	97
4	Клим	Суворов	Львович	113	Теория категории A	80
5	Доминик	Гелетей	Платонович	114	Теория категории B	53
6	Заур	Несвитайло	Иванович	114	Теория категории B	69
7	Остин	Фролов	Владимирович	114	Теория категории B	76
8	Стефан	Новиков	Вадимович	114	Теория категории B	73

Рисунок 5.5.- Результат п’ятого запиту

6) Запит виводить кількість осіб в групах, у які є проблеми з оплатою (з використанням підзапиту):

```
SELECT gr."Номер_группы",
   curs."Категория_курса",
    COUNT (st."Состояние_оплаты_за_курс") AS BillingProblems
FROM "Группа" gr,
   "Курс_обучения" curs,
    "Студент" st
WHERE gr."Курс" = curs."id_курса"
   AND st."Группа" = gr."id_группы"
            AND "Состояние_оплаты_за_курс" = ANY (SELECT
"Состояние_оплаты_за_курс" FROM "Студент" WHERE
"Состояние_оплаты_за_курс" = 'Не уплачено' OR st."Состояние_оплаты_за_курс" =
'Уплачено частично')
GROUP BY gr."Номер_группы", curs."Категория_курса";
```

	Номер_группы integer	Категория_курса character(1)	billingproblems bigint
1	111	B	3
2	112	A	1
3	113	B	2
4	114	B	3
5	115	C	3

Рисунок 5.6.- Результат шостого запиту

7) Запит виводить студентів у віці 18 років і менше в порядку убування їх віку:

```
SELECT ans."Имя",
   ans."Фамилия",
   ans."Отчество",
   ans."Дата_рождения",
   ans."Место_учебы/работы"
FROM "Анкета_студента" ans
WHERE ans."Дата_рождения" BETWEEN '2000-01-01' AND '2018-01-01'
ORDER BY ans."Дата_рождения";
```

	Имя character(20)	Фамилия character(20)	Отчество character(20)	Дата_рождения date	Место_учебы/работы character varying
1	Зуфар	Громов	Сергеевич	2000-03-10	Студент ОНПУ
2	Мирослав	Прохоров	Грегориевич	2000-05-01	Студент МГУ им.Ломоносова
3	Юлий	Гелетей	Вадимович	2000-12-30	Кассир в McDonalds
4	Лев	Многогрешный	Дмитриевич	2001-04-05	Студент КНПУ
5	Иммануил	Павлик	Станиславович	2002-09-03	Студент ОНУ
6	Аким	Дорофеев	Ярославович	2003-06-04	Студент ОНМУ

Рисунок 5.7.- Результат сьомого запиту

8) Запит виводить курсантів, які є студентами будь-яких вузів:

```
SELECT ans."Имя",
   ans."Фамилия",
   ans."Отчество",
   gr."Номер_группы",
   ans."Место_учебы/работы"
FROM "Анкета_студента" ans,
    "Группа" gr,
    "Студент" st
WHERE st."Группа" = gr."id_группы"
  AND ans."id_анкеты" = st."Анкета"
            AND ans."Место_учебы/работы" LIKE '%Студент%'
ORDER BY ans."Дата_рождения";
```

	Имя character(20)	Фамилия character(20)	Отчество character(20)	Номер_группы integer	Место_учебы/работы character varying
1	Марк	Дубченко	Грегориевич	112	Студент ОНУ
2	Орест	Белов	Иванович	113	Студент ОНМА
3	Зуфар	Громов	Сергеевич	111	Студент ОНПУ
4	Мирослав	Прохоров	Грегориевич	111	Студент МГУ им.Ломоносова
5	Лев	Многогрешный	Дмитриевич	111	Студент КНПУ
6	Иммануил	Павлик	Станиславович	111	Студент ОНУ
7	Аким	Дорофеев	Ярославович	113	Студент ОНМУ

Рисунок 5.8.- Результат восьмого запиту

9) Запит, який виводить інформацію про інструкторів і кількість іспитів, які вони прийняли за час своєї роботи:

```
SELECT  ins."Фамилия",
          ins."Имя",
   ins."Отчество",
   COUNT(jer."Инструктор") AS  "Принятых_экзаменов",
   ins."Опыт_работы(в_месяцах)",
   ins."Зарплата"
FROM "Инструктор" ins,
    "Журнал" jer
WHERE jer."Инструктор" = ins."id_инструктора"
GROUP  BY  ins."Имя",  ins."Фамилия",  ins."Отчество",  ins."Опыт_работы(в_месяцах)",
ins."Зарплата"
ORDER BY COUNT(jer."Инструктор");
```

	Фамилия character(20)	Имя character(20)	Отчество character(20)	Принятых_экзаменов bigint	Опыт_работы(в_месяцах) integer	Зарплата double precision
1	Борисов	Тарас	Константинович	4	16	8500
2	Кудрявцев	Роман	Русланович	4	15	8000
3	Леонтьев	Филипп	Эдуардович	5	5	10000
4	Охота	Юрий	Александрович	6	9	9600

Рисунок 5.9.- Результат дев’ятого запиту

10) Запит, який вирощує викладачів потрібної категорії з наявністю приватних занять:

```
SELECT  prof."Фамилия",
   prof."Имя",
    prof."Отчество",
    prof."Категории_обучения"
FROM "Преподаватель" prof
WHERE prof."Категории_обучения" IN ('B', 'A')
      AND prof."Наличие_частных занятий" = true;
```

	Фамилия character(20)	Имя character(20)	Отчество character(20)	Категории_обучения character(10)
1	Бутко	Клаус	Ярославович	B
2	Андреев	Болеслав	Геннадьевич	B
3	Михайлов	Авдей	Тарасович	B

Рисунок 5.10.- Результат дев’ятого запиту

11) Запит, який виводить свідоцтва про закінчень студентів, які закінчили курс в 2017:

```
SELECT ans."Имя",
   ans."Фамилия",
   ans."Отчество",
   gr."Номер_группы",
   log."Дата_прохождения" AS "Дата выпуска",
   log."Набранный_балл",
   curs."Категория_курса"
FROM "Студент" st,
   "Анкета_студента" ans,
   "Журнал" log,
   "Группа" gr,
   "Курс_обучения" curs,
   "Свидетельство_об_окончании" cert,
   "Тип_экзамена" exam
  WHERE st."Анкета" = ans."id_анкеты"
  AND st."Группа" = gr."id_группы"
     AND log."Студент" = st."id_студента"
            AND cert."Студент" =  st."Анкета"
                  AND curs."id_курса" = cert."Пройденный_курс"
                        AND log."Экзамен" = exam."id_экзамена"
                              AND exam."Категория_экзамена" LIKE
'%Теория%'
                                    AND log."Набранный_балл" > 60
GROUP BY ans."Имя", ans."Фамилия", ans."Отчество", gr."Номер_группы",
log."Дата_прохождения", log."Набранный_балл", curs."Категория_курса"
HAVING log."Дата_прохождения" < '2018.01.01';
```

	Имя character(20)	Фамилия character(20)	Отчество character(20)	Номер_группы integer	Дата выпуска date	Набранный_балл integer	Категория_курса character(1)
1	Заур	Несвитайло	Иванович	114	2017-01-29	69	B
2	Остин	Фролов	Владимирович	114	2017-01-26	76	B
3	Стефан	Новиков	Вадимович	114	2017-01-28	73	B

12) Підрахунок суми грошей, які отримала автошкола від студентів за всі роки роботи:

```
SELECT SUM(curs."Стоимость") + SUM(exam."Стоимость") AS "Сумма за обучение"
FROM "Студент" st,
   "Группа" gr,
   "Курс_обучения" curs,
   "Тип_экзамена" exam,
   "Журнал" jer
  WHERE st."Группа" = gr."id_группы"
            AND gr."Курс" = curs."id_курса"
                  AND exam."id_экзамена" = jer."Экзамен"
                              AND jer."Студент" = st."id_студента";
```

	Сумма за обучение double precision
1	126250

6 СТВОРЕННЯ КОРИСТУВАЧІВ І ПРИЗНАЧЕННЯ ПРАВ ДОСТУПУ

Поняття управління доступом має сенс, у першу чергу, у багатокористувальницькому режимі роботи СУБД.

На рівні БД управління доступом полягає в створенні і видаленні користувачів шляхом призначення їхніх імен, паролів і прав доступу до конкретних БД і є прерогативою адміністратора БД (DBA або SYSDBA). Кожен користувач є власником таблиць, пов'язаних з тією або іншою БД. Відповідно, власник таблиці може керувати доступом до даних на рівні таблиць. Таке управління полягає в призначенні привілеїв іншим користувачам, тобто визначення того, може чи ні конкретний користувач виконувати ту або іншу команду.

Привілеї SQL є об'єктними. Тобто користувач має привілей (право) виконувати конкретну команду для конкретного об'єкта БД: таблиці, представлення, стовпця тощо.

Привілеї, призначені власником таблиці, збігаються за мнемонікою та за змістом з командами МВД SQL: SELECT, INSERT, DELETE, UPDATE та REFERENCES.

SQL використовується зазвичай в середовищах, які вимагають розпізнавання користувачів і відмінності між різними користувачами систем. Взагалі кажучи, адміністратори баз даних, самі створюють користувачів і дають їм привілеї. З іншого боку, користувачі, які створюють таблиці, самі мають права на управління цими таблицями. Привілеї – це то, що визначає, чи може вказаний користувач виконати дану команду. Є кілька типів привілеїв, що відповідають декільком типам операцій. привілеї даються і скасовуються двом командами SQL: - GRANT (ДОПУСК) і REVOKE (СКАСУВАННЯ).

Кожен користувач в середовищі SQL, має спеціальне ідентифікаційне ім'я або номер. Команда, послана в базі даних, асоціюється з певним користувачем або інакше, спеціальним ідентифікатором доступу. Оскільки це

відноситься до SQL базі даних, ID дозволу - це ім'я користувача, і SQL може використовувати спеціальне ключове слово USER, яке направляється в ідентифікатора доступу пов'язаного з поточною командою. Команда інтерпретується і дозволяється (або забороняється) на основі інформації пов'язаної з Ідентифікатором доступу користувача, який подав команду.

Потрібно створити двох користувачів, один з них буде добавляти та змінювати оцінки, інший буде мати усі права над таблицею абітурієнтів.

1) create user admin with password 'root';

grant all on "Студент", "Преподаватель", "Курс_обучения", "Журнал", "Заявка_в_автошколу", "Группа", "Анкета_студента", "Свидетельство_об_окончании", "Тип_экзамена", "Инструктор" to admin;

2) CREATE USER lecturer WITH PASSWORD '1234';

GRANT SELECT ON top_theor_exam_log TO lecturer;
GRANT SELECT ON full_info TO lecturer;
GRANT SELECT ON top_prakt_exam_log TO lecturer;
GRANT ALL ON "Группа" TO lecturer;
GRANT SELECT ON "Преподаватель" TO lecturer;
GRANT SELECT ON "Инструктор" TO lecturer;
GRANT ALL ON "Журнал" TO lecturer;
GRANT UPDATE ON "Свидетельство_об_окончании" TO lecturer;

3) CREATE USER student WITH PASSWORD '1111';

GRANT SELECT ON top_theor_exam_log TO student;
GRANT SELECT ON full_info TO student;
GRANT SELECT ON top_prakt_exam_log TO student;
GRANT UPDATE ON "Анкета_студента" TO student;
GRANT UPDATE ON "Заявка_в_автошколу" TO student;
GRANT SELECT ON "Группа" TO student;
GRANT SELECT ON "Журнал" TO student;

4) grant select on "Курс_обучения" to public – можливість усім читати з таблиці "Курс_обучения"

ВИСНОВКИ

В ході виконання курсової роботи за темою : “Розробка бази даних для підтримки діяльності адміністрації автошколи з обліку студентів, які навчаються за різними категоріями в автошколі” були спроектована та створена база даних, яка складається з 10 таблиць та може бути застосована на практиці у реальних закладах.

У роботі представлені 5 тригерів та 3 функції які полегшують та автоматизують процес використання БД, а також 12 прикладів різноманітних запитів отримання інформації з таблиць з використанням підзапитів, агрегатних та групових функцій, з критеріями відбору та використанням спеціальних операторів. Також були реалізовані 4 представлення та об’явлені права користування ними серед користувачів БД. Було створено 3 користувача БД.

Під час виконання роботи були отримані навички проектування бази даних, її реалізація та заповнення її функціоналом.

Завершуючи роботу, можна прийти до висновку, що складання баз даних це робота з великою кількістю тонкощів. А PostgreSQL це дуже потужний інструмент, який може надати велику кількість можливостей для створення БД.

ПЕРЕЛІК ПОСИЛАНЬ ТА ЛІТЕРАТУРИ

1. Конспект лекцій по курсу "Організація баз даних та знань" для студентів спеціальності 122 - "Комп'ютерні науки" / Укладач: М.Г. Глава. Одеса: ОНПУ, 2017.– 68 с
2. PostgreSQL Reference Manual - Volume 1: SQL Language Reference — The PostgreSQL Global Development Group, 2007.
3. Wikipedia[Електронний ресурс] : Вільна бібліотека.- Режим доступу: https://ru.wikipedia.org/wiki/
4. Хабрахабр [Електронний ресурс] : Найоригінальніший IT проект.- Режим доступу: https://habr.com/
5. Малахов Є.В., Блажко О.А., Глава М.Г. Проектування БД та їх реалізація засобами стандартного SQL та PostgreSQL: Навч. посібник для студ. вищих навч. закладів. – О.: ВМВ, 2012.– 248 с
6. Глава М.Г. Організація баз даних та знань: Конспект лекцій [Електронний ресурс].– Режим доступу: http://library.opu.ua.
7. W3schools.com [Електронний ресурс]: THE WORLD'S LARGEST WEB DEVELOPER SIT. - Режим доступу: https://www.w3schools.com/
8. Postgresql.org [Електронний ресурс]: Postgresql официальная документация. - Режим доступу: https://www.postgresql.org/docs/

ДОДАТОК А

«Створення таблиць»

```
CREATE SEQUENCE s_Student;
CREATE SEQUENCE s_Professor;
CREATE SEQUENCE s_StudCourse;
CREATE SEQUENCE s_Journal;
CREATE SEQUENCE s_Application;
CREATE SEQUENCE s_Group;
CREATE SEQUENCE s_Form;
CREATE SEQUENCE s_Certificate;
CREATE SEQUENCE s_ExamType;
CREATE SEQUENCE s_Instructor;

CREATE TABLE "Студент" (
	"id_студента" INT DEFAULT NEXTVAL ('s_Student') UNIQUE,
	"Группа" int NOT NULL,
	"Анкета" int NOT NULL UNIQUE,
	"Количество_пропусков" int NOT NULL,
	"Состояние_оплаты_за_курс" char(30) NOT NULL,
	CONSTRAINT Студент_pk PRIMARY KEY ("id_студента")
);

CREATE TABLE "Преподаватель" (
	"id_преподавателя" INT DEFAULT NEXTVAL ('s_Professor') UNIQUE,
	"Фамилия" char(20) NOT NULL,
	"Имя" char(20) NOT NULL,
	"Отчество" char(20) NOT NULL,
	"Категории_обучения" char(10) NOT NULL,
	"Наличие_частных_занятий" bool NOT NULL,
	"Зарплата" decimal NOT NULL,
	"Адресс_проживения" char(30) NOT NULL,
	"Высшее_образование" char(20) NOT NULL,
	CONSTRAINT Преподаватель_pk PRIMARY KEY ("id_преподавателя")
);
```

```
CREATE TABLE "Курс_обучения" (
        "id_курса" INT DEFAULT NEXTVAL ('s_StudCourse') UNIQUE,
        "Категория_курса" char NOT NULL,
        "Стоимость" int NOT NULL,
        "Продолжительность" int NOT NULL,
        CONSTRAINT Курс_обучения_pk PRIMARY KEY ("id_курса")
);

CREATE TABLE "Журнал" (
        "id_журнала" INT DEFAULT NEXTVAL ('s_Journal') UNIQUE,
        "Студент" int NOT NULL,
        "Экзамен" int NOT NULL,
        "Дата_прохождения" DATE NOT NULL,
        "Набранный_балл" int NOT NULL,
        "Инструктор" int NOT NULL,
        CONSTRAINT Журнал_pk PRIMARY KEY ("id_журнала")
);

CREATE TABLE "Заявка_в_автошколу" (
        "id_заявки" INT DEFAULT NEXTVAL ('s_Application'),
        "Анкета" INT NOT NULL,
        "Необходимая_категория" char(1) NOT NULL,
        CONSTRAINT Заявка_в_автошколу_pk PRIMARY KEY ("id_заявки","Анкета")
);

CREATE TABLE "Группа" (
        "id_группы" INT DEFAULT NEXTVAL ('s_Group') UNIQUE,
        "Номер_группы" int NOT NULL,
        "Количество_учащихся" int NOT NULL,
        "Курс" int NOT NULL,
        "Дата_начала" DATE NOT NULL,
        "Расписание_занятий" char(30) NOT NULL,
        "Преподаватель" int NOT NULL,
        CONSTRAINT Группа_pk PRIMARY KEY ("id_группы")
);
```

```
CREATE TABLE "Анкета_студента" (
        "id_анкеты" INT DEFAULT NEXTVAL ('s_Form'),
        "Фамилия" char(20) NOT NULL,
        "Имя" char(20) NOT NULL,
        "Отчество" char(20) NOT NULL,
        "Дата_рождения" DATE NOT NULL,
        "Адресс_проживания" char(30) NOT NULL,
        "Место_учебы/работы" varchar NOT NULL,
        CONSTRAINT Анкета_студента_pk PRIMARY KEY ("id_анкеты")
);

CREATE TABLE "Свидетельство_об_окончании" (
        "id_свидетельства" INT DEFAULT NEXTVAL ('s_Certificate'),
        "Пройденный_курс" int NOT NULL,
        "Студент" int NOT NULL,
        "Результат_экзамена" int NOT NULL UNIQUE,
        CONSTRAINT Свидетельство_об_окончании_pk PRIMARY KEY
("id_свидетельства")
);

CREATE TABLE "Тип_экзамена" (
        "id_экзамена" INT DEFAULT NEXTVAL ('s_ExamType'),
        "Категория_экзамена" char(20) NOT NULL,
        "Система_оценивания" char(20) NOT NULL,
        "Стоимость" decimal NOT NULL,
        CONSTRAINT Экзамен_pk PRIMARY KEY ("id_экзамена")
);

CREATE TABLE "Инструктор" (
        "id_инструктора" INT DEFAULT NEXTVAL ('s_Instructor'),
        "Фамилия" char(20) NOT NULL,
        "Имя" char(20) NOT NULL,
        "Отчество" char(20) NOT NULL,
        "Опыт_работы(в_месяцах)" int,
        "Опыт_вождения(в_месяцах)" int,
```

```
    "Категория_транспорта" char(10) NOT NULL,
    "Зарплата" decimal,
    CONSTRAINT Инструктор_pk PRIMARY KEY ("id_инструктора")
);

--Надання зв’язків
ALTER TABLE "Студент" ADD CONSTRAINT "Студент_fk0" FOREIGN KEY ("Анкета")
REFERENCES "Анкета_студента"("id_анкеты");
ALTER TABLE "Студент" ADD CONSTRAINT "Студент_fk1" FOREIGN KEY ("Группа")
REFERENCES "Группа"("id_группы");

ALTER TABLE "Группа" ADD CONSTRAINT "Группа_fk1" FOREIGN KEY
("Преподаватель") REFERENCES "Преподаватель"("id_преподавателя");
ALTER TABLE "Группа" ADD CONSTRAINT "Группа_fk0" FOREIGN KEY ("Курс")
REFERENCES "Курс_обучения"("id_курса");

ALTER TABLE "Журнал" ADD CONSTRAINT "Журнал_fk0" FOREIGN KEY ("Студент")
REFERENCES "Студент"("id_студента");
ALTER TABLE "Журнал" ADD CONSTRAINT "Журнал_fk1" FOREIGN KEY
("Инструктор") REFERENCES "Инструктор"("id_инструктора");
ALTER TABLE "Журнал" ADD CONSTRAINT "Журнал_fk3" FOREIGN KEY ("Экзамен")
REFERENCES "Тип_экзамена"("id_экзамена");

ALTER TABLE "Заявка_в_автошколу" ADD CONSTRAINT "Заявка_в_автошколу_fk0"
FOREIGN KEY ("Анкета") REFERENCES "Анкета_студента"("id_анкеты");

ALTER TABLE "Свидетельство_об_окончании" ADD CONSTRAINT
"Свидетельство_об_окончании_fk0" FOREIGN KEY ("Пройденный_курс") REFERENCES
"Курс_обучения"("id_курса");
ALTER TABLE "Свидетельство_об_окончании" ADD CONSTRAINT
"Свидетельство_об_окончании_fk1" FOREIGN KEY ("Студент") REFERENCES
"Студент"("id_студента");
ALTER TABLE "Свидетельство_об_окончании" ADD CONSTRAINT
"Свидетельство_об_окончании_fk2" FOREIGN KEY ("Результат_экзамена")
REFERENCES "Журнал"("id_журнала");
```