ЗМІСТ

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

3

ІС КР 122 АІ-174 П20

Розроб.

Узун І.С.

Перевір.

Глава М.Г.

Реценз.

Н. Контр.

Затверд.

Розробка бази даних для підтримки діяльності адміністрації автошколи з обліку студентів, які навчаються за різними категоріями в автошколі

Літ.

Акрушів

35

ОНПУ, каф. ІС, гр. АІ-174

Вступ……………………………………………………..……………………….. 4

1. Постановка задачі……………………………………………..…………….…5
2. Проектування бази даних…………………………………….…………….…6
3. Вибір програмного забезпечення………………………………………..…..10
4. Створення бази даних………………………………………...……….…......11
   1. Створення таблиць…………………………….……...…………....11
   2. Створення представлень……………………....……...…………....12
   3. Створення тригерів………………………………………..……….15
   4. Створення функцій……………………………………..……….….20
5. Маніпулювання даними…...…………………………………………..….….22
6. Створення користувачів і призначення прав доступу…….…………….…31

Висновки………………………………………………………………………33

Перелік посилань……………………………………………………………..34

Додаток А……………………………………………………………………..35

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.

# 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 ї практичного застосування у автошколах, що значно полегшить та прискорить работу з ведення обліку курсантів.

# 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) B.**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) B.**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.

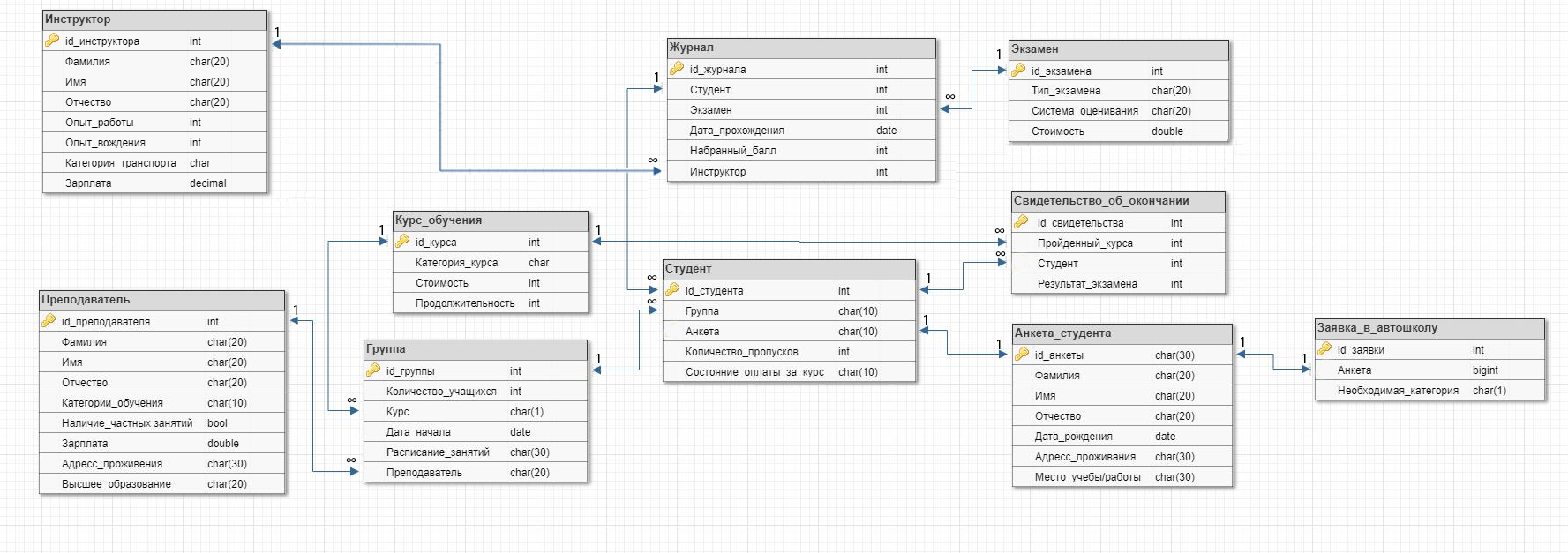


Рисунок 2.1 - Схема даних

# 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";

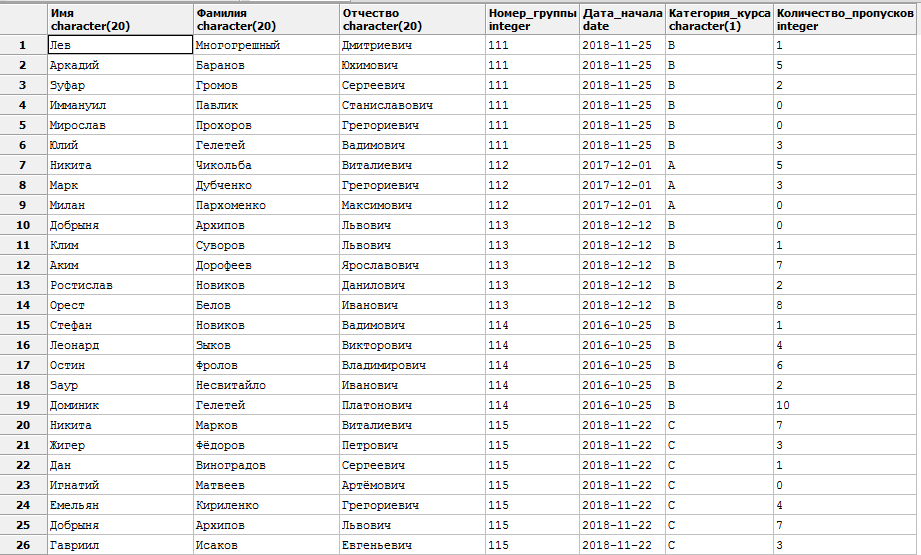


Рисунок 4.1 – Представлення «full\_info»

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;



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;



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";

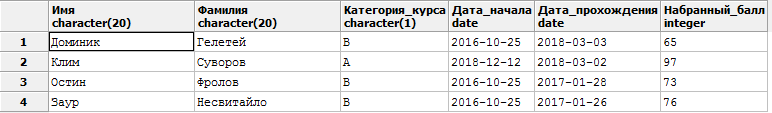


Рисунок 4.4 – Представлення «certificates»

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();

1. 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();

1. 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" = 'С'

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 сhangeExamMark(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';

1. 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';

1. 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';

1. 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 абітуріентів:

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;

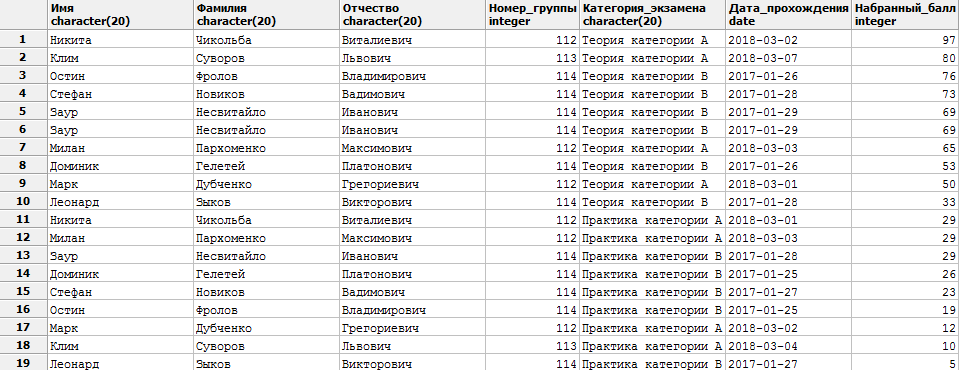


Рисунок 5.1.- Результат першого запиту

1. 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";

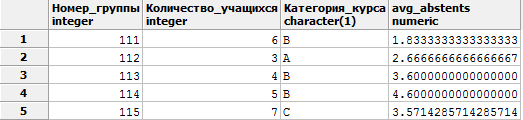


Рисунок 5.2.- Результат другого запиту

1. 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";

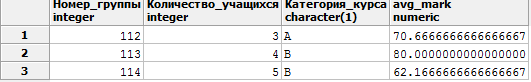


Рисунок 5.3.- Результат третього запиту

1. 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";

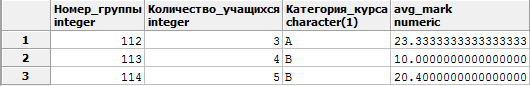


Рисунок 5.4.- Результат четвертого запиту

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");

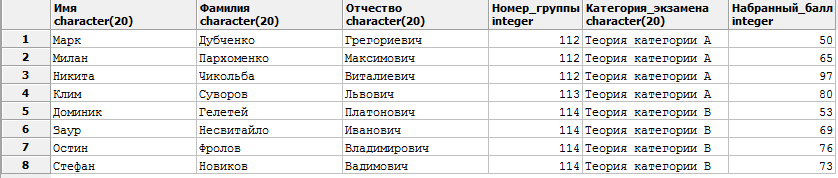


Рисунок 5.5.- Результат п’ятого запиту

1. 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";



Рисунок 5.6.- Результат шостого запиту

1. 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";

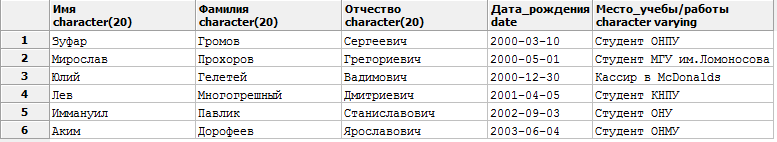


Рисунок 5.7.- Результат сьомого запиту

1. 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";

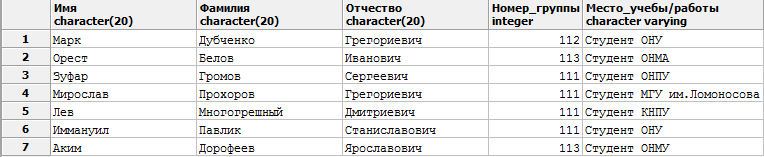


Рисунок 5.8.- Результат восьмого запиту

1. 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");



Рисунок 5.9.- Результат дев’ятого запиту

1. 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;

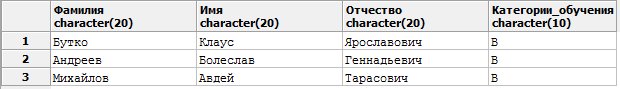


Рисунок 5.10.- Результат дев’ятого запиту

1. 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"

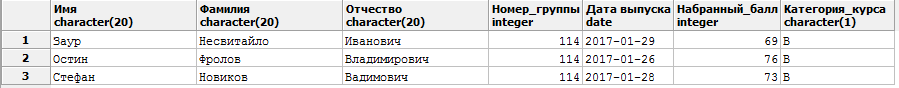
AND log."Экзамен" = exam."id\_экзамена"

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';



1. 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";



1. 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 адмініст-ратора 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 таб-лиць. Таке управління полягає в призначенні привілеїв іншим користувачам, тобто визна-чення того, може чи ні конкретний користувач виконувати ту або іншу команду.

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

Привілеї, призначені власником таблиці, збігаються за мнемонікою та за змістом з ко-мандами 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;

1. 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;

1. 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;

1. 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");