ЗМІСТ

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

3

ІС КР 122 АІ-174 ПЗ1

Розроб.

Ахмад Р.Ш.

Перевір.

Глава М.Г.

Реценз.

Н. Контр.

Затверд.

Розробка бази даних абітурієнтів для автоматизації роботи приймальної комісії ВНЗ

Літ.

Акрушів

28

ОНПУ, каф. ІС, гр. АІ-174

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

1. Постановка задачі……………………………………………..…………….…5
2. Проектування бази даних…………………………………….…………….…6
3. Вибір програмного забезпечення………………………………………..……9
4. Створення бази даних………………………………………...……….…......10
   1. Створення таблиць…………………………….……...…………....10
   2. Створення представлень……………………....……...…………....11
   3. Створення тригерів………………………………………..……….13
   4. Створення функцій……………………………………..……….….16
5. Маніпулювання даними…...…………………………………………..….….18
6. Створення користувачів і призначення прав доступу…….…………….…24

Висновки………………………………………………………………………25

Перелік посилань……………………………………………………………..26

Додаток А……………………………………………………………………..27

ВСТУП

Потоки інформації, що циркулюють у світі, що нас оточує, величезні. З часом вони мають тенденцію до збільшення. Тому у будь-якій організації, як великий, так і малій, виникає проблема організації управління даними, яка забезпечила б найбільш ефективну роботу. І вже сьогодні без баз даних неможливо уявити роботу фінансових, торгівельних та інших організацій.

Віртуальний світ — це уявний світ, для якого визначені певні фізичні та математичні закони і який, при відповідних умовах, може бути за допомогою технічних засобів зроблений доступним для сприйняття органами почуттів людини.

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 необхідної інформації в організації. 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 створена база даних, яка відповідає за автоматизацію роботи приймальної комісії ВНЗ. Також база даних зберігає можливість аналізувати дані абітурієнтів що подають документи до вузу.

Створена база даних вирішує питання формування великої кількості інформації з якою можно зустрітися при створенні автоматизованих систем розрахування оцінок.

Використання цієї бази даних зробить працю з великою кількістю інформації набагато легше. Також працівники не повинні своїми руками описувати, складати рейтинг абітурієнтів, за них це може зробити база даних.

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.

Таким чином вказано, що ця база даних повинна зберігати мінімум інформації о вузах, але максимум інформації про абітурієнтів.

# ПРОЕКТУВАННЯ БАЗИ ДАНИХ

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

Опис сутностей:

1. Сутність «Enrollee» - містить в собі інформацію о абітурієнтах поступаючих до вуза.
2. Сутність «Address» - містить в собі інформацію про адресу абітурієнта.
3. Сутність «Exam» - містить в собі інформацію про здані іспити.
4. Сутність «Education» - містить інформацію про навчальні заклади з яких прийшли абітурієнти.
5. Сутність «Privilegii» - містить інформацію про пільги
6. Сутність «Subject» - містить набір предметів з яких можливе складання ЗНО
7. Сутність «Specialty» - містить інформацію про те, на які спеціальності можливо вступити.
8. Сутність «Reward» - містить інформацію про винагороди абітурієнтів

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

Таблиця 2.1 – Таблиці та їх атрібути

|  |  |  |  |
| --- | --- | --- | --- |
| Ім'я сутності | Назва трибуту | Тип даних | Ключ |
| Enrollee | Id\_kod | int | Первинний |
| Name | char |  |
| Surname | char |  |
| SecName | char |  |
| Id\_privileges | int | Зовнішній |
| birthday | date |  |
| Id\_education | int | Зовнішній |
| Id\_address | int | Зовнішній |
| Id\_reward | int | Зовнішній |
| Subject | Id\_subject | int | Первинний |
| Subject | char |  |
| Education | Id\_education | int | Первинний |
| Institution | char |  |
| Number | int |  |
| City | char |  |
| Exam | Id\_exam | int | Первинний |
| Id\_kod | int | Зовнішній |
| Subject | int |  |
| Mark | int |  |
| Privilegii | Id\_privileges | int | Первинний |
| Privilegii | char |  |
| Reward | Id\_reward | int | Первинний |
| Reward | char |  |
| Address | Id\_address | int | Первинний |
| Index | int |  |
| City | char |  |
| Street | char |  |
| House | char |  |
| Number | int |  |
| Telephon | char |  |
| Specialty | Id\_spec | int | Первинний |
| Namespec | char |  |

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)

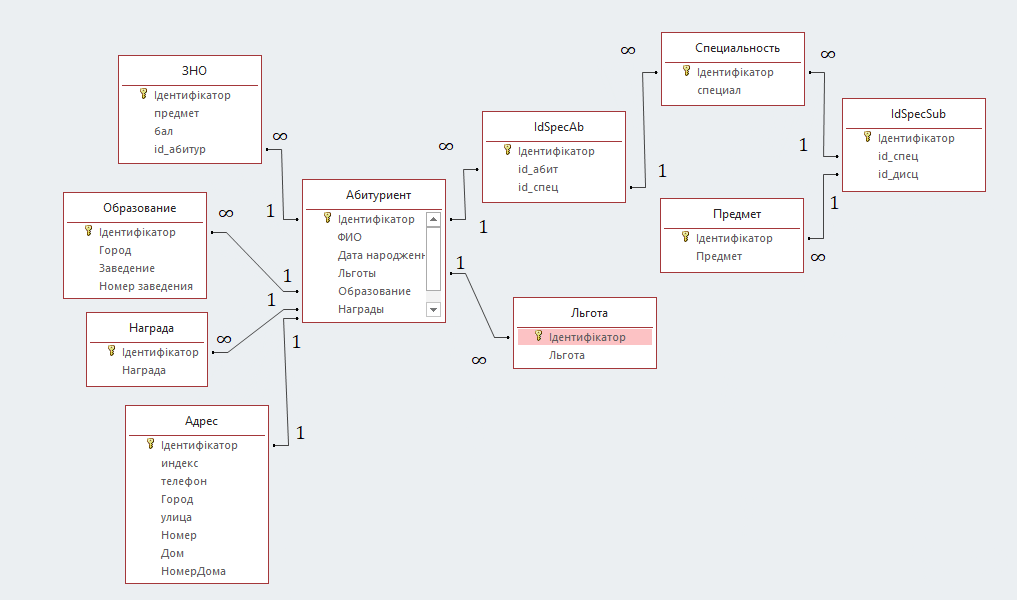


Рисунок 2.1 - Схема даних

# ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

При виконанні наданого завдання була використана Система Управління Базами Даних PostgreSQL. Version 9.5 for 64 Bit Windows Platforms.

PostgreSQL не просто реляційна, а об'єктно-реляційна СУБД. Це дає йому деякі переваги над іншими SQL базами даних з відкритим вихідним кодом, такими як MySQL, MariaDB і Firebird.

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.  
 Якщо раптом так трапиться, що великого списку типів даних Постгреса вам виявиться недостатньо, ви можете використовувати команду CREATE TYPE, щоб створити нові типи даних, такі як складової, що перераховується, діапазон і базовий.

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

# 4 СТВОРЕННЯ БАЗИ ДАНИХ

4.1 Створення таблиць

Настройка таблиць це дуже тонка робота, існує величезна кількість обмежень, які можна вказувати на всю таблицю або на певні стовпці, іноді доводиться дробити таблиці на таблиці по-менше або навпаки - збирати дрібні таблиці воєдино, частина синтаксису вказано нижче:

Для створення таблиць:  
CREATE TABLE им’я\_таблиці (  
{ им’я \_поля тип\_даних [ DEFAULT значення по замовчуванню ] [ обмеження поля ]}  
[,{…}]  
обмеження таблиці  
);  
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 створення таблиці винагород:

create sequance s\_reward;

create table reward

( id\_reward INT PRIMARY KEY DEFAULT NEXTVAL ('s\_reward'),

reward varchar);

4.2 Створення представлень

Представлення (VIEW) - об'єкт даних який не містить ніяких даних його власника. Це - тип таблиці, чиє утримання вибирається з інших таблиць за допомогою виконання запиту. Оскільки значення в цих таблицях змінюються, то автоматично, їх значення можуть бути показані представленнями.

Ці таблиці виходять з інших таблиць. Вони працюють в запитах і операторах DML точно також як і основні таблиці, але не містять ніяких власних даних. Представлення - подібні вікнам, через які ви переглядаєте інформацію, яка фактично зберігається в базовій таблиці. Представлення - це фактично запит, який виконується всякий раз, коли уявлення стає темою ко манди. Щоб пристрій запитував при цьому в кожен момент стає змістом уявлення.

1. Створення представлення  для перегляду всієї інформації про абітурієнтів

CREATE OR REPLACE VIEW public.allinfo AS

SELECT A.surname,A.name,A.secname,A.birthday,pr.privileges, ad.index,

ad.city,ad.street,ad.house,ad.number,ad.telephon,

ed.institution,ed.number,r.reward

FROM enrollee a, address ad ,education ed, privilegii pr, reward r

WHERE a.id\_address = ad.id\_address AND a.id\_education = ed.id\_education AND a.id\_privileges = pr.id\_privileges AND a.id\_reward = r.id\_reward

ORDER BY a.surname;

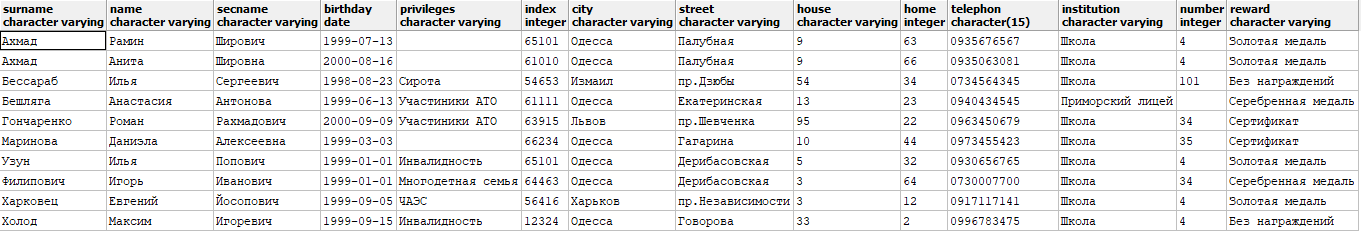


Рисунок 4.1 – Представлення «allinfo»

1. Створення представлення для визначення кращих 10 абітурієнтів.

CREATE OR REPLACE VIEW public.top10 AS

SELECT a.surname,a.name,a.secname,avg(ex.mark) AS avg

FROM enrollee a,exam ex

WHERE a.id\_kod = ex.id\_kod

GROUP BY a.surname, a.name, a.secname

ORDER BY (avg(ex.mark)) DESC

LIMIT 10;



Рисунок 4.2 – Представлення «top10»

1. Створення представлення для того, щоб передати можливість ,

CREATE OR REPLACE VIEW public.enrolllewithpriv AS

SELECT a.surname,a.name,a.secname, a.birthday,pr.privileges

FROM enrollee a,privilegii pr

WHERE a.id\_privileges = pr.id\_privileges

ORDER BY a.surname;



Рисунок 4.3 – Представлення « enrolllewithpriv »

4.3.Створення триггерів

Тригер - збережена процедура особливого типу, яку користувач не викликає безпосередньо, а виконання якої 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.)

Тригер, який видаляє адресу абітурієнта, якщо був видалений абітурієнт:

CREATE FUNCTION DelEnrollee()

RETURNS trigger

AS

$$BEGIN

DELETE FROM address ad where ad.id\_address=old.id\_address;

END; $$

LANGUAGE 'plpgsql';

CREATE TRIGGER DelEnr

BEFORE DELETE ON enrollee

FOR EACH ROW

EXECUTE PROCEDURE DelEnrollee();

Тригер створений для перевірки того, щоб оцінки не була більша за 200 балів.

REATE FUNCTION checkmark()

RETURNS trigger

AS

$$ BEGIN

IF new.mark>200 THEN

RAISE EXCEPTION 'Mark cant be more than 200';

ELSE

return new;

END IF;

END; $$

LANGUAGE 'plpgsql';

CREATE TRIGGER checkNoMore200

BEFORE INSERT ON exam

FOR EACH ROW

EXECUTE PROCEDURE checkmark();

Тригер, який слідкує за тим, щоб номера телефонів не повторювались:

Create function insertAdress() returns trigger as $$

begin

if new.telephon is null then

raise exception 'it cant be null';

end if;

if new.telephon in (select telephon

from address where address.telephon=new.telephon) then

raise exception 'anyone cant take one phone twice';

end if;

return new;

end;

$$ language plpgsql;

create trigger insertAdress before insert or update on address

for each row execute procedure insertAdress();

Тригер, який при видаленні абітурієнта , видалить у інших таблицях його код.

CREATE OR REPLACE FUNCTION delidspec()

RETURNS trigger AS

$$

begin

delete from idspecsub ss where ss.id\_spec=old.id\_spec;

delete from idspecab sc where sc.id\_spec=old.id\_spec;

return old;

end;

$$

LANGUAGE 'plpgsql';

CREATE TRIGGER delspecandsub

BEFORE DELETE

ON public.specialty

FOR EACH ROW

EXECUTE PROCEDURE public.delidspec();

4.4.Створення функцій

Функції є блоками коду, що 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:

Функція, яка змінює в певного абітурієнта оцінку за певний предмет

CREATE FUNCTION changeAllMark(int, int ,int)

RETURNS varchar

AS $$

BEGIN

UPDATE exam SET mark=$3 where subject=$2 AND id\_kod=$1;

return (select surname from enrollee where id\_kod=$1);

END;

$$

LANGUAGE 'plpgsql';

Функція, яка лікує і виводить сред.оцінку певного абітурієнта

CREATE FUNCTION AVGmarkEnrById (int)

RETURNS varchar

AS $$

BEGIN

return (select AVG(mark)

from exam e where e.id\_kod=$1);

END;

$$

LANGUAGE 'plpgsql';

Функція, який дозволяє змінити id в табл Privilegii та в залежних таблицях

create function changeIdPrivilegii(was int, become int) returns integer as $$

begin

if become not in (select id\_kod from enrollee) then

raise exception 'this engaged';

end if;

update enrollee set id\_privileges= null where id\_privileges=was;

update privilegii set id\_privileges= become where id\_privileges=was;

update enrollee set id\_privileges= become where id\_privileges IS null;

return 1;

end;

$$ language plpgsql;

1. МАНІПУЛЮВАННЯ ДАНИМИ

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.

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:

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 A.surname,A.name,A.secname,A.birthday,pr.privileges, ad.index,

ad.city,ad.street,ad.house,ad.number,ad.telephon,

ed.institution,ed.number,r.reward

from enrollee a, address ad ,education ed, privilegii pr, reward r

where a.id\_address=ad.id\_address

AND a.id\_education=ed.id\_education

AND a.id\_privileges=pr.id\_privileges

AND a.id\_reward=r.id\_reward

ORDER BY a.surname

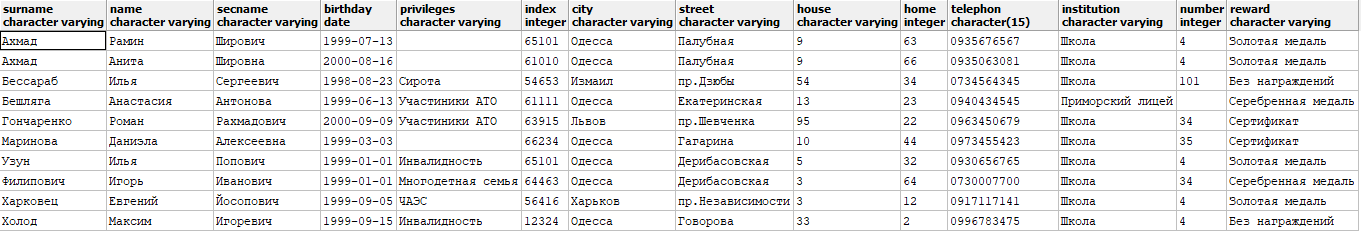


Рисунок 5.1.- Результат першого запиту

1. Запит, який виводить абітурієнтів та спеціальність на яку вони поступають

select spec.namespec,A.id\_kod,A.surname,A.name,A.secname,A.birthday

from specialty spec, enrollee a ,idspecab isa

where a.id\_kod=isa.id\_kod AND spec.id\_spec=isa.id\_spec

group by spec.namespec,a.id\_kod,a.surname

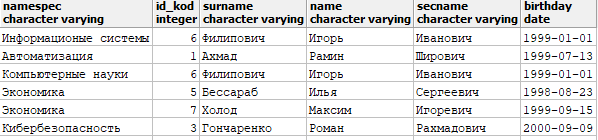


Рисунок 5.2.- Результат другого запиту

1. Запит, який виводить які предмети треба складати на спеціальності

select spec.namespec,sub.subject

from specialty spec, subject sub, idspecsub iss

where spec.id\_spec=iss.id\_spec AND sub.id\_subject=iss.id\_subject

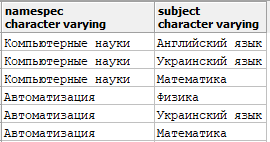


Рисунок 5.3. - Результат третього запиту

1. Запит, який виводить абітуріентів ,які приїхали з інших міст

select a.surname,a.name, a.secname,ad.city

from enrollee a,address ad

where a.id\_address=ad.id\_address AND NOT(ad.city='Одесса')

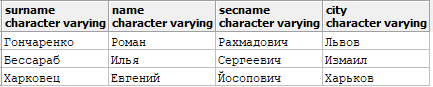


Рисунок 5.4.- Результат четвертого запиту

1. Запит, який виводить інформацію про пільги абітуріентів

select A.surname,A.name,A.secname,A.birthday,pr.privileges

from enrollee a, privilegii pr

where a.id\_privileges=pr.id\_privileges

ORDER BY a.surname

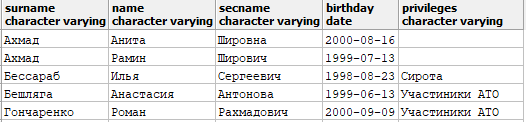


Рисунок 5.5.- Результат п’ятого запиту

1. Запит, який виводить кращих 10 абітуріентів

select surname,name,secname, AVG(mark)

from enrollee a,exam ex

where a.id\_kod=ex.id\_kod

group by surname,name,secname

order by AVG(mark) desc

limit 10



Рисунок 5.6.- Результат шостого запиту

1. Запит, який виводить інформацію про те , хто які предмети ЗНО здавав іх оцінки

select sub.subject,mark,surname,name,secname

from enrollee a,subject sub,exam ex

where a.id\_kod=ex.id\_kod and sub.id\_subject=ex.subject

order by ex.subject,mark desc

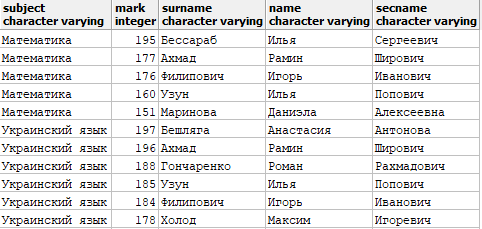


Рисунок 5.7.- Результат сьомого запиту

1. Запит який виводить хто поступає з одеських учбових закладів

select surname,name,secname

from enrollee a

where id\_education IN( select id\_education from education where city ='Одесса')

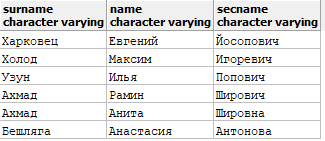


Рисунок 5.8.- Результат восьмого запиту

1. Запит, який виводить кількість нагород кожної школи

select institution,number,reward, COUNT(a.id\_reward)

from education ed,reward r,enrollee a

where a.id\_education=ed.id\_education AND a.id\_reward = r.id\_reward

group by institution,number,reward

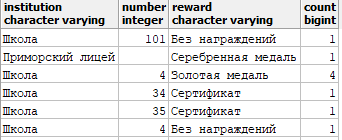


Рисунок 5.9.- Результат дев’ятого запиту

1. Запит, який виводить працівників, які постают з льготою і не мешкають в Одесі

select a.surname ,a.name

from enrollee a

where id\_address IN

(select id\_address from address where city != 'Одесса')

AND id\_privileges = ANY

( select id\_privileges from privilegii)



Рисунок 5.10.- Результат десятого запиту

1. СТВОРЕННЯ КОРИСТУВАЧІВ І ПРИЗНАЧЕННЯ ПРАВ ДОСТУПУ

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 adinuser with password '1234';

grant insert, select, update on exam to adinuser;

1. create user enrolleeuser with password '1111';

grant all on enrollee to enrolleeuser;

1. grant select on enrollee to public – можливість усім читати з таблиці enrollee
2. create user adminn with password '1307'

grant all on enrollee,exam,subject,specialty,idspecab,idspecsub,address,privilegii,reward to adminn

# ВИСНОВКИ

Завершуючи роботу, можна прийти до висновку, що складання баз даних це складна робота з великою кількістю тонкощів.

Результатом виконання курсової роботи став розроблений додаток баз даних, дозволяючий автоматизувати роботу с даними абітурієнтів.

Протягом виконання курсової роботи були отримані навички проектування бази даних, її реалізація та заповнення її функціоналом.

PostgreSQL це дуже потужний інструмент, який може надати велику кількість можливостей, більшість з яких, нажаль, не були використані протягом виконання курсової роботи.

# 

# 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://habrahabr.ru/>
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. Гаврилова Т.А. Базы знаний интеллектуальных систем: учеб.пособие / Т.А. Гаврилова, В.Ф. Хорошевский.– СПб.: Питер, 2001.– 384 с. (31 шт)

# ДОДАТОК А

# «Створення таблиць»

# Створення таблиці винагород:

create sequance s\_reward;

create table reward

( id\_reward INT PRIMARY KEY DEFAULT NEXTVAL ('s\_reward'),

reward varchar);

Створення таблиці учбових закладів

create sequence s\_education

create table education

( id\_education INT PRIMARY KEY DEFAULT NEXTVAL ('s\_education'),

institution varchar,

number int,

city varchar);

Створення таблиці пільг

create sequance s\_privileges;

create table privileges

( id\_privileges INT PRIMARY KEY DEFAULT NEXTVAL ('s\_privileges'),

privileges varchar);

Створення таблиці адрес

create sequance s\_address;

create table address

( id\_address INT PRIMARY KEY REFERENCES enrollee(id\_kod)

index int,

city varchar,

street varchar,

house varchar,

number int,

telephon char(15));

Створення таблиці абітурієнтів

create sequance s\_enrollee;

create table enrollee

( id\_kod INT PRIMARY KEY DEFAULT NEXTVAL ('s\_enrollee'),

name varchar,

surname varchar,

secName varchar,

birthday date,

id\_privileges INT REFERENCES privileges(id\_privileges),

id\_education INT REFERENCES education(id\_education),

id\_reward INT REFERENCES reward(id\_reward));

id\_address INT REFERENCES enrollee (id\_enrollee) UNIQUE

);

Створення таблиці для складового ключа між таблицями спеціальностей і абітурієнтів

create sequance s\_IdSpecAb;

create table IdSpecAb

( id\_kod INT REFERENCES enrollee (id\_kod),

id\_spec INT REFERENCES speialty(id\_spec)

PRIMARY KEY (id\_kod,id\_spec)

);

Створення таблиці спеціальностей

create sequance s\_specialty;

create table specialty

( id\_spec INT PRIMARY KEY DEFAULT NEXTVAL ('s\_specialty'),

nameSpec varchar);

Створення таблиці для складового ключа між таблицями спеціальностей і предметів ЗНО

create sequance s\_IdSpecSub;

create table IdSpecSub

( id\_spec INT REFERENCES specialty(id\_spec),

id\_subject INT REFERENCES subject (id\_subject),

PRIMARY KEY (id\_spec,id\_subject)

);

Створення таблиці предметів ЗНО

create sequance s\_subject;

create table subject

( id\_subject INT PRIMARY KEY DEFAULT NEXTVAL ('s\_subject'),

subject varchar);

Створення таблиці результатів іспитів

create sequance s\_exam

create table exam

( id\_exam INT PRIMARY KEY DEFAULT NEXTVAL ('s\_exam'),

subject INT REFERENCES subject (id\_subject)

mark INT,

id\_kod INT REFERENCES enrollee (id\_kod)

);