Міністерство освіти і науки України

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

Кафедра інженерії програмного забезпечення (№ 603)

**Навчально-методичне забезпечення дисципліни**

***Бази даних***

**Галузь знань:** 12 Інформаційні технології

**Спеціальність:** 121 Інженерія програмного забезпечення

**Освітня програма:** Інженерія програмного забезпечення

**Рівень вищої освіти**: перший (бакалаврський)

Розробник: Манжос Ю.С., доц.,к.т.н

(прізвище та ініціали, посада, науковий ступінь і вчене звання)

Харків

2019

**ЗМІСТ**

Конспект лекцій 3

Змістовий модуль 1. Реляційна модель даних. 3

Змістовий модуль 2. Основи мови SQL 3

Змістовий модуль 3. Основи мови PL/SQL 4

Змістовий модуль 4. Розосереджені бази даних 4

Методичні вказівки до виконання лабораторних робіт 5

Лабораторна робота № 1 Вивчення команд DDL. 5

Висновки: 13

Лабораторна робота № 4 Створення програмних пакетів PL/SQL, робота з курсорами 20

Лабораторна Робота № 5 Тригери баз даних 26

Лабораторна робота № 6 Побудова ER-моделі предметної області. 33

Лабораторна робота № 7 Побудова логічної моделі бази даних. 37

Лабораторна робота № 8 Побудова фізичної моделі даних. 41

Лабораторна робота № 9 Розробка SQL-скриптів запитів, зберігаємих процедур та функцій, тригерів. 46

Тематика індивідуальних завдань 59

Методичні вказівки до самостійної роботи студентів 60

Питання, тести для контрольних заходів 62

Змістовий модуль 1: 62

Змістовий модуль 2: 63

Змістовий модуль 3: 64

Змістовий модуль 4: 65

Питання для самоперевірки 67

Рекомендована література, інформаційні ресурси 70

Базова 70

Допоміжна 70

Інформаційні ресурси 70

# Конспект лекцій

1. Реляційна модель даних.
	1. Вступ. Історія розвитку БД. Основні задачі БД. Етапи проектування БД. Моделі даних, застосовні на різних етапах. Реляційна модель даних. Концептуальна, логічна, фізична моделі даних. Основні елементи моделей даних. Різниці між моделями даних. Сутності та їх атрибути. Моделювання сутностей відношеннями. Моделювання відношень таблицями. Елементи відношень. Елементи таблиць. Мова створення елементів баз даних. Типи даних. Послідовності (лічильники). Створення бази даних, таблиць, стовпчиків та зв’язків. Відтворення, внесення, модифікація та видалення даних з таблиць бази даних
	2. Проектування бази даних. Визначення вимог до системи. Збір і аналізування вимог користувачів. Проектування бази даних. Нормалізація даних як шлях проектування БД. Перша, друга та треті нормальні форми. Четвнерта нормальна форма та її властивості. Пята нормальна форма та її використання. Приведеня моделі даних до шостої нормпальної форми. Вибір цільової системи керування базою даних. Властивості систем керування базами даних. Розроблення додатків. Створення прототипів. Реалізація. Конвертування і завантаження даних. Тестування бази даних. Експлуатація і супровід баз даних.
	3. Відношення та їх властивості. ER-метод проектування. Трансформація ER-моделі в реляційну модель даних. Обмеження цілісності реляційної моделі даних. Види цілісності моделей даних. Підтримка цілісності баз даних. Підтримка цілісності сутностей та зв’язків. Денормалізація даних та ре факторинг бази даних для підвищення її продуктивності. Шифрування. Контрзаходи і некомп’ютерні засоби контролю доступу до даних. Підходи до забезпечення безпеки і планування захисту від непередбачених обставин.
2. Основи мови SQL
	1. Мова DDL та її основи. Основні типи даних. Реалізація даних в MySQL. Мова SQL. Мова визначення даних DDL SQL. Схема даних. Типи даних, що підтримує система керування БД. Таблиці, стовпчики таблиць, подання, обмеження, ключ, зв’язки та особливості їх створення мовою DDL. Кардинальність та призначення зв’язків. Рольові зв’язки. Звязки між різними таблицями. Рекурсивні зв’язки. Звязки тип-підтип. Звязки типу 1:1, 1:К, К:М та їх використання у моделях даних.
	2. Мова DML. Маніпулювання даними DML SQL, відтворення даних. Послідовності. Перевірка умов. Запити до даних. Команда добування даних SELECT. Однотабличні запити, запити з умовами, запити на об'єднання даних. Корельвані запити. Запити з об’єднанням таблиць. Використання аліасів стовпчиків та таблиць у запитах до баз даних.
	3. Складні запити до бази даних. Багатотабличні запити, кореляційні запити, запити з агрегацією. Агрегація з умовою. Агрегатні функції та їх використання для проектування баз даних. Агрегація з аласами. Способи підвищення продуктивності агрегованих запитів.
	4. Елементи об'єктно-орієнтованого програмування у базах даних. Об'єктні типи у МySQL. Розміщення об'єктів у базі даних. Використання об'єктних типів. Геоінформаційні типи даних. Створення та використання об’єктних типів під час проектування бази даних. Запити з об’єктними типами даних.
3. Основи мови PL/SQL
	1. Процедури у MySQL. Основи PL/SQL. Блоки. Структура блока. Оголошення змінних. Типи даних у мові PL/SQL. Скалярні і складені типи (записи та таблиці). Використання SQL у коді PL/SQL. Вбудовані функції PL/SQL. Процедури та функції мовою PL/SQL та їх написання. Завантаження процедур у БД. Виконання процедур та функцій. Параметри в процедурах та функціях.
	2. Курсори у БД як засіб пришвидшення доступу до баз даних. Типи курсорів. Реалізація курсорів у процедурах PL/SQL. Особливості синтаксису курсорів. Способи використання курсорів для збірання певних даних у процедурах та функціях.
	3. Тригери у БД. Призначення тригерів. Створення тригерів. Типи тригерів. Тригери для операція вставки, модифікації та видаленя записів з таблиць. Тригери до та після виконання операцій з модифікації записів. Використання функцій та процедур у тригерах. Способи оптимізації тригерів. Реалізація каскадного відновлення даних у реляційних базах даних
4. Розосереджені бази даних
	1. Захист даних. Транзакції, фіксація та відкат. Цілісність даних. Моделі транзакцій. Відновлення бази даних. Поліпшені моделі транзакцій. Модель вкладених транзакцій. Керування транзакціями. Підтримка транзакцій. Створення транзакцій мовою TCL. Реалізація транзакцій мовами TCL та PL/SQL. Відкат транзакцій. Штатне завершення транзакцій.
	2. Розроблення розосереджених баз даних. Архітектури баз даних. Файл-серверна архітектура, клієнт-серверна архітектура, три гривнева архітектура. Розподілені бази даних та їх використання. Концепції і розроблення розподілених СУБД. Принципи організації комп'ютерних мереж. Функції й архітектура розподілених СУБД. Керування доступом до баз даних. Привілеї, користувачі і ролі. Мова DCL як засіб контролю привілеями та користувачами. Визначення привілеїв, ролей та користувачів у базах даних.

# Методичні вказівки до виконання лабораторних робіт

## Лабораторна робота № 1 Вивчення команд DDL.

*Мета роботи:* Ознайомитися з основними можливостями створення елементів фізичної бази даних .

### ***Завдання***

1. Відповідно до варіанту створити таблиці та інші об’єкти БД (обов’язковість атрибутів визначити самостійно).
2. Для таблиць визначити всі обмеження цілісності.
3. Перевірити коректність визначених обмежень
4. Заповнити кожну з таблиць 10 – 20 записами. Для додавання даних у таблиці як первиний ключ використати вказані послідовності
5. Скласти звіт

### **Зміст звіту**

1. Зміст.
2. Постановка задачі.
3. Результат – програмний код та копії екрану
4. Висновки.

**Завдання для виконання лабораторної роботи №1**

ВАРІАНТ 1. Банківські рахунки

|  |  |  |
| --- | --- | --- |
| Стовпчик таблиці | Тип даних | Опис |
| Ідентифікаційний код клієнта | число | Первинний ключ |
| Прізвище | рядок |  |
| Імя | рядок |  |
| По-батькові | рядок |  |
| Адреса | рядок |  |
| Телефон | рядок | Форма рядка ХХХ-ХХ-ХХДе Х- цифра |
| Банківські рахунки |
| Стовпчик таблиці | Тип даних | Опис |
| Номер банківського рахунку | число | Первинний ключ |
| Ідентифікаційний код клієнта | число | Зовнішній ключ |
| Тип рахунку | рядок | Одно зі значень: зарплатний або депозитний |
| Валюта | рядок | Одно зі значень: грн., дол., євро |
| Дата створення рахунку | дата | Поточна дата |

 Всі інші варіанти для всіх робіт є у «Лабораторному практикуму. Бази даних. Основи мови SQL». Харків, ХАІ, 2014 р., 83 стор.

Приклад виконання лабораторної роботи №1

План виконання роботи

Створити таблиці та інші об'єкти БД, зазначені в варіанті завдання (обов'язковість атрибутів задати самостійно).

Для заданих таблиць визначити всі обмеження цілісності.

Перевірити правильність роботи створених обмежень.

Заповнити кожну з таблиць 10 - 20 записами. При вставці даних для генерації первинного ключа необхідно використовувати зазначені послідовності.

Оформити звіт.

Варіант №6

Оплата комунальних платежів

|  |
| --- |
| Види платежів |
| Поле | Тип поля | Опис |
| Номер платежу | Число | Первинний ключ |
| Назва | Рядок |  |
| Оплата в місяць за одну людину, грн | Число | Дійсне число (два знаки після коми) |
| Мешканці |
| Поле | Тип поля | Опис |
|  | Число | Первинний ключ |
| Особистий рахунок | Число |  |
| Прізвище | Рядок |  |
| Ім’я | Рядок |  |
| По батькові | Рядок |  |
| К-ть прописаних людей | Число |  |
| Пільги, % | Число | Приймає значення від 0 до 30 |
| **Мешканець оплатив комунальний платіж** |
| Поле | Тип поля | Опис |
| Номер квитанції | Число | Первинний ключ |
| Дата оплати  | Дата | За замовчуванням поточна дата |
| Особистий рахунок | Число | Зовнішній ключ |
| Номер платежу  | Число | Зовнішній ключ |
| Оплачено, грн | Число | Дійсне число (два знаки після коми) |

Хід виконання роботи

1. Була створена база даних «Оплата комунальних платежів»:

CREATE SCHEMA `оплата комунальних платежів` ;

Були створені 3 таблиці, згідно з умовою у варіанті:

Види платежів:

CREATE TABLE `види платежів` (

 `Номер платежу` INT NOT NULL AUTO\_INCREMENT COMMENT '',

 `Назва` VARCHAR(100) NOT NULL COMMENT '',

 `Оплата в місяць за одну людину, грн` DECIMAL(10,2) NOT NULL COMMENT '',

 PRIMARY KEY (`Номер платежу`) COMMENT '');

Мешканці:

CREATE TABLE `мешканці` (

 `Особистий рахунок` int(11) NOT NULL AUTO\_INCREMENT,

 `Прізвище` varchar(70) COLLATE utf8mb4\_unicode\_ci NOT NULL,

 `Імя` varchar(60) COLLATE utf8mb4\_unicode\_ci NOT NULL,

 `По батькові` varchar(65) COLLATE utf8mb4\_unicode\_ci NOT NULL,

 `К-ть прописаних людей` int(11) DEFAULT NULL,

 `Пільги, %` decimal(10,2) DEFAULT NULL CHECK (`Пільги, %` > 0 and `Пільги, %` < 30),

 PRIMARY KEY (`Особистий рахунок`));

Мешканець оплатив комунальний платіж:

CREATE TABLE `мешканець оплатив комунальний платіж` (

 `Номер квитанції` int(11) NOT NULL AUTO\_INCREMENT,

 `Дата оплати` date NOT NULL DEFAULT Now(),

 `Особистий рахунок` int(11) NOT NULL,

 `Номер платежу` int(11) NOT NULL,

 `Оплачено, грн` decimal(10,2) NOT NULL,

 PRIMARY KEY (`Номер квитанції`),

 KEY `Номер платежу\_idx` (`Номер платежу`),

 KEY `Особистий рахунок\_idx` (`Особистий рахунок`),

 CONSTRAINT `Номер платежу` FOREIGN KEY (`Номер платежу`) REFERENCES `види платежів` (`Номер платежу`),

 CONSTRAINT `Особистий рахунок` FOREIGN KEY (`Особистий рахунок`) REFERENCES `мешканці` (`Особистий рахунок`));

В кожну з таблиць було додано записи. Для перегляду записів були створені наступні запити:

SELECT \*

FROM `оплата комунальних платежів`.`види платежів`;

Результат зображено на рис. 3.1.



Рисунок 3.1 – Записи таблиці «Види платежів»

SELECT \*

FROM `оплата комунальних платежів`.`мешканці`;

Результат зображено на рис. 3.2.



Рисунок 3.2 – Записи таблиці «Мешканці»

SELECT \*

FROM `оплата комунальних платежів`.`мешканець оплатив комунальний платіж`;

Результат зображено на рис. 3.3.



Рисунок 3.3 – Записи таблиці «Мешканець оплатив комунальний платіж»

Висновки:

В результаті виконання лабораторної роботи була створена база даних, відповідно до варіанта. Створені таблиці. Для таблиць були визначені всі обмеження цілісності. Кожна таблиця була заповнена даними, результат представлений на рисунках 1-3.

**Лабораторна робота № 2**. Однотабличні запити на зчитування, зміну та додавання інформації.

Мета роботи: отримати навички створення запитів, що зчитують, змінюють та додають інформацію до реляційної БД.

Зміст звіту

1. Ознайомитися з теорією створення запитів.
2. Розробити та виконати запити, зазначені в вашому варіанті.
3. Скласти та реалізувати свої три однотабличні запити.
4. Перевірити правильність роботи створених запитів.
5. Оформити звіт.

Завдання

1. Реалізувати запити, які повертають всі рядки і всі стовпці кожної зі створених в лабораторній роботі №1 таблиць.
2. Виконати запити, зазначені в вашому варіанті.
3. Скласти та реалізувати свої три однотабличні запити.
4. Перевірити правильність роботи створених запитів.
5. Оформити звіт.

Приклад виконання лабораторної роботи № 2

Варіант №6

Запити, які повертають всі рядки і всі стовпці кожної зі створених в лабораторній роботі №1 таблиць.

Запит, що повертає всі рядки та стовбці таблиці Види платежів, результат якого зображено на рисунку 1.1:

SELECT \* FROM `види платежів`;



Рисунок 1.1 – Записи таблиці Види платежів

Запит, що повертає всі рядки та стовбці таблиці Мешканці, результат якого зображено на рисунку 1.2:

SELECT \* FROM мешканці;



Рисунок 1.2 – Записи таблиці Види платежів

Запит, що повертає всі рядки та ствобці таблиці Мешканець оплатив комунальний платіж, результат якого зображено на рисунку 1.3:

SELECT \* FROM `мешканець оплатив комунальний платіж`;



Рисунок 1.3 – Записи таблиці Види платежів

1. Запити мого варіанту:

Вивести алфавітний список прізвищ та ініціалів мешканців з номерами особових рахунків, відсортоване за відсотком знижки.

SELECT

`Особистий рахунок`,

concat(`Прізвище`,' ',left(`Імя`,1),'. ',left(`По батькові`,1),'.') as ПІБ,

`Пільги, %`

FROM `мешканці`

Order by `Пільги, %`, `Прізвище`;

Результат виконання запиту зображено на рисунку 2.1:



Рисунок 2.1 – Алфавітний список прізвищ та ініціалів мешканців з номерами особових рахунків, відсортоване за відсотком знижки.

Вивести значення максимального, мінімального, середнього платежів і загальну суму всіх платежів.

SELECT

 MAX(`Оплачено, грн`) as 'Максимальна оплата платежу',

 MIN(`Оплачено, грн`) as 'Минимальная оплата платежу',

 Round(AVG(`Оплачено, грн`),2) as 'Середня оплата платежу',

 SUM(`Оплачено, грн`) as 'Загальна сума оплати платежів'

FROM `мешканець оплатив комунальний платіж`;

Результат виконання запиту зображено на рисунку 2.2:



Рисунок 2.2 – Значення максимального, мінімального, середнього платежів і загальна сума всіх платежів

Вивести загальну суму платежів, внесену мешканцями за N-й місяць.

SELECT

 SUM(`Оплачено, грн`) AS 'Загальна сума платежів'

 FROM `мешканець оплатив комунальний платіж`

WHERE Month(`Дата оплати`)= 9;

Результат виконання запиту зображено на рисунку 2.3:



Рисунок 2.3 – Загальна сума платежів, внесена мешканцями за N-й (9й) місяць

Вивести номери тих платежів, сумарна оплата за які в N-му місяці склала більше M грн.

SELECT `Номер платежу`

FROM `мешканець оплатив комунальний платіж`

WHERE month(`Дата оплати`) = 10

GROUP BY `Номер платежу`

HAVING SUM(`Оплачено, грн`) > 200

Результат виконання запиту зображено на рисунку 2.4:



Рисунок 2.4 – Номери тих платежів, сумарна оплата за які в N-му (10му) місяці склала більше M (200) грн.

Для особового рахунку N і номера платежу M вивести останню суму оплати.

SELECT \*

FROM `мешканець оплатив комунальний платіж`

Where (`Особистий рахунок`=5 and `Номер платежу`=2)

ORDER BY `Дата оплати` desc

LIMIT 1

Результат виконання запиту зображено на рисунку 2.5:



Рисунок 2.5 – Остання сума оплати для особового рахунку N (5) і номера платежу M (2)

Мої три однотабличні запити:

Вивести загальну суму оплати для кожного особового рахунку:

select `Особистий рахунок`, sum(`Оплачено, грн`) as 'Всього оплачено'

from `мешканець оплатив комунальний платіж`

group by `Особистий рахунок`

Результат виконання запиту зображено на рисунку 3.1:



Рисунок 3.1 – Загальна сума оплати для кожного особового рахунку

Кількість людей, що внесли оплату в місяці M (10) за Nй(2) вид платежу.

SELECT Count(\*) as 'Кількість людей', month(`Дата оплати`) as 'Місяць оплати', `Номер платежу`

FROM `мешканець оплатив комунальний платіж`

WHERE (month(`Дата оплати`) = 10 and `Номер платежу`=2)

Результат виконання запиту зображено на рисунку 3.2:



Рисунок 3.2 – Кількість людей, що внесли оплату в місяці M (10) за Nй(2) вид платежу

Вивести рейтинг, який складається з 3х найдорожчих платежів.

SELECT Назва, `Оплата, грн`

FROM `види платежів`

ORDER BY `Оплата, грн` desc

LIMIT 3

Результат виконання запиту зображено на рисунку 3.3:



Рисунок 3.3 – Рейтинг з 3х найдорожчих платежів

Висновки:

В результаті виконання лабораторної роботи було створено та виконано 5 запитів в базі даних, створеній в лабораторній роботі №1. Запити створювались за умовою в варіанті №6. Також було складено 3 однотабличних запити.

Лабораторна Робота № 3 Багатотабличні запити, зовнішнє з’єднання таблиць, вкладені підзапити

Мета роботи: отримання навичок створення багато табличних запитів до баз даних.

### **Зміст звіту**

1. Опис варіанту.
2. Програмний код.
3. Копії екранів, що підтверджують виконання.
4. Блок-схеми алгоритмів.
5. Висновки.

### **Завдання**

1. Знайти декартовий добуток двох (трьох) таблиць, створених в лабораторній роботі №1, пояснити отриману кількість рядків в результуючій таблиці.
2. Виконати запити, вказані в варіанті (кожен пункт завдання реалізується одним оператором Select).
3. Скласти та реалізувати свої три багатотабличні запити.
4. Перевірити правильність роботи створених запитів.
5. Оформити звіт.

Приклад виконання лабораторної роботи № 3

Варіант №6

1. Запит для знаходження декартового добутку таблиць Мешканці та «Мешканець оплатив комунальний платіж», результат якого зображено на рисунку 1.1:

SELECT мешканці.\*, `мешканець оплатив комунальний платіж`.\*

FROM мешканці, `мешканець оплатив комунальний платіж`



Рисунок 1.1 – Записи таблиці Види платежів

В отриманій таблиці 80 рядків. Ця кількість зумовлена тим, що кожен рядок таблиці Мешканці (8 рядків) з’єднався з кожним рядком таблиці «Мешканець оплатив комунальний платіж» (10 рядків), в добутку маємо 80 рядків, хоча більшість рядків – «інформаційний шум».

1. Запити мого варіанту:

Запит, що виводить загальну суму заданого по назві платежу, внесеного мешканцями за останній квартал.

SELECT SUM(`Оплачено, грн`) AS 'Загальна сума',

QUARTER(`мешканець оплатив комунальний платіж`.`Дата оплати`) AS Квартал

FROM `види платежів`,`мешканець оплатив комунальний платіж`

WHERE `види платежів`.Назва='Опалення'

AND `види платежів`.`Номер платежу`=`мешканець оплатив комунальний платіж`.`Номер платежу`

AND QUARTER(`мешканець оплатив комунальний платіж`.`Дата оплати`)=QUARTER(NOW())

Результат виконання запиту зображено на рисунку 2.1:



Рисунок 2.1 – Загальна сума заданого по назві платежу (Опалення), внесеного мешканцями за останній квартал

Запит, що виводить для мешканця з особистим рахунком N назву та дату платежів, які були зроблені за останній місяць, а також номера відповідних квитанцій.

SELECT `Особистий рахунок`, Назва AS 'Назва платежу', `Дата оплати`,

`Номер квитанції`

FROM `види платежів`,`мешканець оплатив комунальний платіж`

WHERE `види платежів`.`Номер платежу`=`мешканець оплатив комунальний платіж`.`Номер платежу`

AND MONTH(`Дата оплати`)=MONTH(NOW()) AND `Особистий рахунок`=5

Результат виконання запиту зображено на рисунку 2.2:



Рисунок 2.2 – Назва та дата платежів, які були зроблені за останній місяць, а також номера відповідних квитанцій для мешканця з особистим рахунком 5

Вивести назву тих платежів, сумарна оплата за які в N-му місяці склала більше M грн.

SELECT `види платежів`.Назва AS 'Назва платежу',

MONTH(`Дата оплати`) AS Місяць, SUM(`Оплачено, грн`) AS 'Сумарна оплата'

FROM `види платежів`, `мешканець оплатив комунальний платіж`

WHERE `види платежів`.`Номер платежу`=`мешканець оплатив комунальний платіж`.`Номер платежу` and MONTH(`Дата оплати`)=9

GROUP BY `мешканець оплатив комунальний платіж`.`Номер платежу`

HAVING SUM(`Оплачено, грн`) > 200

Результат виконання запиту зображено на рисунку 2.3:



Рисунок 2.3 – Назви тих платежів, сумарна оплата за які в N-му (9му) місяці склала більше M (200) грн.

Для кожного мешканця вивести число платежів, які були ним зроблені за останній місяць (якщо їх немає, то повинен виводитись 0).

SELECT мешканці.`Особистий рахунок`,

CONCAT(`Прізвище`,' ',LEFT(`Імя`,1),'. ',LEFT(`По батькові`,1),'.') AS ПІБ,

COUNT(`Мешканці оплата`.`Особистий рахунок`) AS 'Число платежів'

FROM мешканці LEFT JOIN

(SELECT \* FROM `мешканець оплатив комунальний платіж`

WHERE MONTH(`мешканець оплатив комунальний платіж`.`Дата оплати`)=MONTH(NOW())) `Мешканці оплата`

ON мешканці.`Особистий рахунок`=`Мешканці оплата`.`Особистий рахунок`

GROUP BY `Мешканці оплата`.`Особистий рахунок`, мешканці.`Особистий рахунок`

Результат виконання запиту зображено на рисунку 2.4:



Рисунок 2.4 – Число платежів, які були ним зроблені кожним мешканцем за останній місяць

Для кожного мешканця вивести вид останнього платежу та його суму.

select мешканці.`Особистий рахунок`,

concat(мешканці.`Прізвище`,' ',left(мешканці.`Імя`,1),'. ',left(мешканці.`По батькові`,1),'.') as ПІБ, `Платежі`.Назва as `Вид платежу`, `Платежі`.`Дата оплати` as `Дата платежу`, `Платежі`.`Оплачено, грн` as `Сума платежу`

from мешканці left join (select MK.`Особистий рахунок`, VP.Назва, MK.`Оплачено, грн`, MK.`Дата оплати`

from `мешканець оплатив комунальний платіж` MK join `види платежів` VP on MK.`Номер платежу`=VP.`Номер платежу`

where ((select `Номер квитанції`

 from `мешканець оплатив комунальний платіж` MO

 where (MO.`Особистий рахунок`=MK.`Особистий рахунок`)

 order by `Дата оплати` desc limit 1)=MK.`Номер квитанції`)) `Платежі`

 on мешканці.`Особистий рахунок`=`Платежі`.`Особистий рахунок`

Результат виконання запиту зображено на рисунку 2.5:



Рисунок 2.5 –Вид останнього платежу та його сума для кожного мешканця

Мої три багатотабличні запити:

Вивести загальну суму оплати, кількість платежів та назву для кожного виду платежу:

select VP.Назва, count(\*) as `Кількість платежів`, sum(MP.`Оплачено, грн`) as `Всього оплачено`

from `види платежів` VP left join `мешканець оплатив комунальний платіж` MP on VP.`Номер платежу`=MP.`Номер платежу`

group by VP.`Номер платежу`

Результат виконання запиту зображено на рисунку 3.1:



Рисунок 3.1 – Загальна сума оплати, кількість платежів та назва для кожного виду платежу

Вивести для кожного мешканця загальну суму оплати по платежу з номером N.

select мешканці.`Особистий рахунок`, concat(мешканці.`Прізвище`,' ',left(мешканці.`Імя`,1),'. ',left(мешканці.`По батькові`,1),'.') as ПІБ,

`Платежі`.Назва as `Вид платежу`, sum(`Платежі`.`Оплачено, грн`) as `Загальна сума оплати`

from мешканці left join

(select MK.`Особистий рахунок`, VP.Назва, MK.`Оплачено, грн`, MK.`Дата оплати`

from `мешканець оплатив комунальний платіж` MK join `види платежів` VP on MK.`Номер платежу`=VP.`Номер платежу`

where VP.`Номер платежу`=2) `Платежі`

on мешканці.`Особистий рахунок`=`Платежі`.`Особистий рахунок`

group by мешканці.`Особистий рахунок`

Результат виконання запиту зображено на рисунку 3.2:



Рисунок 3.2 – Загальна сума оплати по платежу з номером N (2)

Вивести список мешканців та їх загальну суму оплати платежів за попередній місяць:

select concat(мешканці.`Прізвище`,' ',left(мешканці.`Імя`,1),'. ',left(мешканці.`По батькові`,1),'.') as ПІБ, sum(`Платежі`.`Оплачено, грн`) as `Загальна сума оплати`, month(Платежі.`Дата оплати`) as `Місяць`

from мешканці join `мешканець оплатив комунальний платіж` Платежі on мешканці.`Особистий рахунок`=Платежі.`Особистий рахунок`

group by мешканці.`Особистий рахунок`, month(Платежі.`Дата оплати`)

having month(now())-`Місяць`=1

Результат виконання запиту зображено на рисунку 3.3:



Рисунок 3.3 – Список мешканців та їх загальна суму оплати платежів за попередній місяць

Висновки:

В результаті виконання лабораторної роботи було створено та виконано 1 запит на пошук декартового добутку двох таблиць та 5 запитів за умовою в варіанті №6. Також було складено 3 багатотабличних запити. Запити були створені за допомогою зовнішніх з’єднань таблиць та вкладених підзапитів.

## Лабораторна робота № 4 Створення програмних пакетів PL/SQL, робота з курсорами

Мета роботи: отримання навичок використання курсорів у простих процедурах та функціях мовою PL/SQL.

### **Зміст звіту**

1. Постановка задачі.
2. Програмний код процедур та функцій
3. Копії екрана, що дітверджують виконання.
4. Висновки

### **Завдання**

1. Визначити склад заголовку та тіла пакету.
2. Заголовок пакету обов’язково повинен включати в себе указані в варіанті процедури та функції, а також визначення курсорів з параметрами, побудованих по будь-яким двом операторам Select з попередньої роботи.
3. Для кожної таблиці з первинним ключем, який генерується послідовністю, пакет повинен містити функцію, яка в якості параметра приймає значення полів запису, за винятком первинного ключа. Ця функція повинна отримувати з послідовності новий первинний ключ, вставляти нову запис в таблицю з цим ключем та зі вказаними у вигляді параметрів значеннями полів. Функція повинна повертати первинний ключ вставленої записи або NULL, якщо запис не була вставлена.
4. Тіло пакета, окрім реалізації процедур та функцій, оголошених в заголовку, може додатково включати допоміжні підпрограми.
5. Створити та налагодити пакет (всі процедури повинні передбачати обробку аномальних ситуацій).
6. Перевірити коректність курсорів.

Приклад виконання лабораторної рооти № 4

Умова варіанту №6

Пакет повинен включати в себе:

1. процедуру, яка за номером особистого рахунку мешканця та назві платежу за поточною датою оформлює квитанцію про оплату;
2. функцію, яка за номером особистого рахунку через кому виводить номера оплачених квитанцій за певний період часу, а також виводить їх суму;

функцію, яка для конкретного періоду часу через кому виводить назву всіх платежів та загальну суму оплати за ними.

Хід виконання роботи

Склад таблиць:

Таблиця «Види платежів»:

SELECT \* FROM `види платежів`;

Результат зображено на рисунку 1:



Рисунок 1 – Дані таблиці «Види платежів»

Таблиця «Мешканець оплатив комунальний платіж»:

SELECT \* FROM `мешканець оплатив комунальний платіж`;

Результат зображено на рисунку 2:



Рисунок 2 – Дані таблиці «Мешканець оплатив комунальний платіж»

Таблиця «Мешканці»:

SELECT \* FROM мешканці;

Результат зображено на рисунку 3:



Рисунок 3 – Дані таблиці «Мешканці»

Створення необхідних функцій та процедур

1. Код процедури:

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `Оформити квитанцію`(№Рахунку int(11), Назва\_Платежу varchar(45))

BEGIN

 declare Результат varchar(500) default "";

 declare Мешканець varchar(195) default "";

 declare `К-ть\_Людей` int(11) default 0;

 declare №Платежу int(11) default 0;

 declare Пільги decimal(10,2) default 0;

 declare Оплата decimal(10,2) default 0;

 declare ДоОплати decimal(10,2) default 0;

 declare Мешканці Cursor for SELECT concat(мешканці.`Прізвище`,' ',left(мешканці.`Імя`,1),'. ',left(мешканці.`По батькові`,1),'.') as ПІБ, `К-ть прописаних людей`, `Пільги, %` FROM оплата\_комунальних\_платежів.мешканці where `Особистий рахунок`=№Рахунку;

 declare Види\_платежів Cursor for SELECT `Оплата, грн`, `Номер платежу` FROM оплата\_комунальних\_платежів.`види платежів` where Назва=Назва\_Платежу;

 open Мешканці;

 Fetch Мешканці into Мешканець,`К-ть\_Людей`, Пільги;

 close Мешканці;

 open Види\_платежів;

 Fetch Види\_платежів into Оплата, №Платежу;

 close Види\_платежів;

 set ДоОплати=`К-ть\_Людей`\*Оплата\*(1-Пільги/100);

 insert `мешканець оплатив комунальний платіж` set `Дата оплати`=Now(), `Особистий рахунок`=№Рахунку,

 `Номер платежу`=№Платежу, `Оплачено, грн`=ДоОплати;

END$$

DELIMITER ;

Результат роботи процедури зображено на рисунку 4 для вхідних даних:

CALL `оплата\_комунальних\_платежів`.`Оформити квитанцію`(7, 'Каналізація');



Рисунок 4 – Результат роботи процедури

Код функції:

DELIMITER $$

CREATE DEFINER=`root`@`localhost` FUNCTION `Список квитанцій`(№Рахунку int(11), Початок\_Періоду DATE, Кінець\_Періоду DATE) RETURNS varchar(200) CHARSET cp1251

BEGIN

 declare done int default 0;

 declare result varchar(200) default "";

 declare №Квитанції int(11) default 0;

 declare Сума\_Оплати decimal(10,2) default 0;

 declare Платежі Cursor for select `Номер квитанції`, `Оплачено, грн` from `мешканець оплатив комунальний платіж` where (`Особистий рахунок`=№Рахунку) and (`Дата оплати` BETWEEN Початок\_Періоду AND Кінець\_Періоду);

 DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done=1;

 set result=Concat("Період з ",Початок\_Періоду," по ",Кінець\_Періоду,": ");

 open Платежі;

 while done=0 do

 fetch Платежі into №Квитанції,Сума\_Оплати;

 if done=0 then

 set result=Concat(result,"№",№Квитанції," - ",Сума\_Оплати," грн.; ");

 end if;

 End while;

 close Платежі;

RETURN result;

END$$

DELIMITER ;

Запуск та результат роботи функції зображено на рисунку 5:



Рисунок 5 – Результат роботи функції

Код функції:

DELIMITER $$

CREATE DEFINER=`root`@`localhost` FUNCTION `Список платежів`(Початок\_Періоду DATE, Кінець\_Періоду DATE) RETURNS varchar(200) CHARSET cp1251

BEGIN

 declare done int default 0;

 declare result varchar(200) default "";

 declare Назва\_Платежу varchar(45) default "";

 declare Сума\_Оплати decimal(10,2) default 0;

 declare Дата\_Оплати Date;

 declare Платежі Cursor for

 select VP.Назва, sum(MP.`Оплачено, грн`) as `Сума оплати`, MP.`Дата оплати`

from `види платежів` VP join `мешканець оплатив комунальний платіж` MP on VP.`Номер платежу`=MP.`Номер платежу`

where (MP.`Дата оплати` BETWEEN Початок\_Періоду AND Кінець\_Періоду)

group by VP.Назва;

 DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done=1;

 set result=Concat("Період з ",Початок\_Періоду," по ",Кінець\_Періоду,": ");

 open Платежі;

 while done=0 do

 fetch Платежі into Назва\_Платежу,Сума\_Оплати,Дата\_Оплати;

 if done=0 then

 set result=Concat(result,Назва\_Платежу," - ",Сума\_Оплати," грн.; ");

 end if;

 End while;

 close Платежі;

RETURN result;

END$$

DELIMITER ;

Запуск та результат роботи функції зображено на рисунку 6:



Рисунок 6 – Результат роботи функції

Висновки:

В результаті виконання лабораторної роботи було розроблено процедури та функції, що зберігаються, згідно з умовою варіанту. Також були проведені їх виклики та перевірки на працездатність.

## Лабораторна Робота № 5 Тригери баз даних

Мета роботи: отримання навичок використання тригерів для роботи з базою даних.

**Зміст** звіту

1. Постановка задачі.
2. Програмний код
3. Копії екрану, що підтверджують виконання програми
4. Висновок

Завдання

1. Створити тригери, які повинні виконувати задачі, вказані в варіанті.
2. Для кожної таблиці, яка має первинний ключ, що генерується послідовністю, створити тригер для автоматичної генерації цього первинного ключа при вставці.
3. Здійснити денормалізацію однієї з таблиць, додавши в неї підсумкове поле та забезпечити коректність інформації в цьому полі за допомогою тригеру.
4. За допомогою тригерів до однієї з таблиць додати журнал, в якому повинні бути зафіксовані відомості про певні дії користувачів над цією таблицею.
5. Протестувати створені тригери.
6. Оформити звіт.

Приклад виконання лабораторної роботи №5

Умова варіанту №6

Створити тригер, який буде перевіряти оплату комунальних платежів; якщо мешканець оплатив після 15го числа звітного місяця, то йому начисляється пеня в розмірі 0,5% за кожний прострочений день.

**Тригер, виконуючий певну функцію, задану варіантом**

Перевірка повинна відбуватись при додаванні або при зміні записів, а отже тригер буде створено для оновлення та додавання записів.

Код тригеру на додавання:

DELIMITER $$

CREATE DEFINER=`root`@`localhost`

TRIGGER мешканець\_оплатив\_комунальний\_платіж\_Before\_Insert

BEFORE INSERT ON `мешканець оплатив комунальний платіж` FOR EACH ROW

BEGIN

 declare Дата\_оплати Date;

 declare Кількість\_днів int(11);

 declare Сума\_оплати decimal(10,2);

 set Дата\_оплати=new.`Дата оплати`;

 set Сума\_оплати=new.`Оплачено, грн`;

 if Day(Дата\_оплати)>15 then

 set new.Пеня=0.005\*Сума\_оплати\*(Day(Дата\_оплати)-15);

 END IF;

end$$

Код тригеру на оновлення:

DELIMITER $$

CREATE DEFINER=`root`@`localhost`

TRIGGER мешканець\_оплатив\_комунальний\_платіж\_Before\_Update

BEFORE UPDATE ON `мешканець оплатив комунальний платіж` FOR EACH ROW

BEGIN

 declare Дата\_оплати Date;

 declare Кількість\_днів int(11);

 declare Сума\_оплати decimal(10,2);

 set Дата\_оплати=new.`Дата оплати`;

 set Сума\_оплати=new.`Оплачено, грн`;

 if Day(Дата\_оплати)>15 then

 set new.Пеня=0.005\*Сума\_оплати\*(Day(Дата\_оплати)-15);

 END IF;

end$$

Для перевірки тригеру виконаємо наступний запит:

INSERT INTO ` мешканець оплатив комунальний платіж` (`Дата оплати`, `Особистий рахунок`, `Номер платежу`, `Оплачено, грн`) VALUES ('2019-10-22 00:00:00', '6', '3', '209');

Результат запиту зображено на рисунку 1:



Рисунок 1 – Приклад вставки запису, для якого начисляється пеня

В базі даних, розробленій в першій лабораторній робот, ключі з послідовностями є в трьох таблицях: мешканці, види платежів, мешканець оплатив комунальний платіж.

Код тригеру на BEFORE INSERT для таблиці мешканці:

DELIMITER $$

CREATE DEFINER=`root`@`localhost` TRIGGER мешканці\_BEFORE\_INSERT BEFORE INSERT ON `мешканці` FOR EACH ROW

BEGIN

 declare inc int(10) default 1;

 select `Особистий рахунок` into inc

 from `мешканці`

 order by `Особистий рахунок` desc limit 1;

 set new.`Особистий рахунок`=inc+1;

end$$

Код тригеру на BEFORE INSERT для таблиці види платежів:

DELIMITER $$

CREATE DEFINER=`root`@`localhost`

TRIGGER Види\_платежів\_BEFORE\_INSERT BEFORE INSERT ON `види платежів` FOR EACH ROW

BEGIN

 declare inc int(10) default 1;

 select `Номер платежу` into inc

 from `види платежів`

 order by `Номер платежу` desc limit 1;

 set new.`Номер платежу`=inc+1;

end$$

Код тригеру на BEFORE INSERT для таблиці мешканець оплатив комунальний платіж:

DELIMITER $$

CREATE DEFINER=`root`@`localhost`

TRIGGER Види\_платежів\_BEFORE\_INSERT BEFORE INSERT ON ` мешканець оплатив комунальний платіж ` FOR EACH ROW

BEGIN

 declare inc int(10) default 1;

 select `Номер квитанції` into inc

 from `мешканець оплатив комунальний платіж`

 order by `Номер квитанції` desc limit 1;

 set new.`Номер квитанції`=inc+1;

end$$

Ці тригери знаходять останній за ключем запис та додають до ключа новоствореного запису значення+1.

**Денормалізація таблиці та підсумкове поле**

Додам в таблицю Мешканці підсумкове поле, яке буде містити в собі кількість оплачених платежів з таблиці Мешканець оплатив комунальний платіж.

Код запиту для зміни таблиці:

Alter table мешканці add Column Кількість оплачених платежів INT(11);

Тепер додам 2 тригера на додавання та видалення даних для таблиці Мешканець оплатив комунальний платіж.

Код тригеру на додавання даних:

DELIMITER $$

CREATE DEFINER=`root`@`localhost` trigger перевірка\_кільк\_платежів\_AI

after insert on `мешканець оплатив комунальний платіж`

for each row

begin

 declare done int default 0;

 declare №Рахунку int (11);

 declare Кількість\_Платежів int (11);

 declare Платежі Cursor for

 Select `Особистий рахунок`, count(\*) Кількість from `мешканець оплатив комунальний платіж`

 group by `Особистий рахунок`;

 DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done=1;

 open Платежі;

 while done=0 do

 fetch Платежі into №Рахунку,Кількість\_Платежів;

 if done=0 then

 UPDATE мешканці SET `Кількість оплачених платежів` = Кількість\_Платежів

 WHERE `Особистий рахунок`=№Рахунку;

 end if;

 End while;

 close Платежі;

end$$

Код тригеру на видалення даних:

DELIMITER $$

CREATE DEFINER=`root`@`localhost` trigger перевірка\_кільк\_платежів\_AD

after delete on `мешканець оплатив комунальний платіж`

for each row

begin

 declare done int default 0;

 declare №Рахунку int (11);

 declare Кількість\_Платежів int (11);

 declare Платежі Cursor for

 Select `Особистий рахунок`, count(\*) Кількість from `мешканець оплатив комунальний платіж`

 group by `Особистий рахунок`;

 DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done=1;

 open Платежі;

 while done=0 do

 fetch Платежі into №Рахунку,Кількість\_Платежів;

 if done=0 then

 UPDATE мешканці SET `Кількість оплачених платежів` = Кількість\_Платежів

 WHERE `Особистий рахунок`=№Рахунку;

 end if;

 End while;

 close Платежі;

end$$

Для перевірки тригеру виконаємо наступний запит:

DELETE FROM `оплата\_комунальних\_платежів`.`мешканець оплатив комунальний платіж` WHERE `Номер квитанції`='42';

Результат запиту зображено на рисунку 2:



Рисунок 2 – Всі записи таблиці мешканець оплатив комунальний платіж

Після виконання цього запису в таблиці спрацює тригер та змінить обчислить підсумкове поле в таблиці Мешканці. Записи цієї таблиці зображено на рисунку 3:



Рисунок 3 - Всі записи таблиці Мешканці

**Фіксування дій через тригери в іншу таблицю**

Створюю для таблиці мешканець оплатив комунальний платіж 3 тригера (видалення, оновлення, додавання), які будуть заносити данні в таблицю Журнал\_Операцій: Користувач, Дата\_Операції, Операція, Значення\_ключа.

Код запиту на створення таблиці:

CREATE TABLE `журнал\_операцій` (

 `Користувач` varchar(50) DEFAULT NULL,

 `Дата\_операції` datetime DEFAULT NULL,

 `Операція` varchar(50) DEFAULT NULL,

 `Значення\_ключа` int(10) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=cp1251;

Тепер додаю тригери для таблиці мешканець оплатив комунальний платіж для кожної BEFORE події:

Код для тригеру Insert before:

DELIMITER $$

CREATE DEFINER=`root`@`localhost`

TRIGGER мешканець\_оплатив\_комунальний\_платіж\_Before\_Insert BEFORE INSERT ON `мешканець оплатив комунальний платіж` FOR EACH ROW

BEGIN

insert into журнал\_операцій values( current\_user(), now(), 'додавання запису', new.`Номер квитанції`);

end$$

Код для тригеру update before:

DELIMITER $$

CREATE DEFINER=`root`@`localhost`

TRIGGER мешканець\_оплатив\_комунальний\_платіж\_Before\_Update BEFORE UPDATE ON `мешканець оплатив комунальний платіж` FOR EACH ROW

BEGIN

insert into журнал\_операцій values( current\_user(), now(), 'змінення запису', new.`Номер квитанції`);

end$$

Код для тригеру delete before:

DELIMITER $$

CREATE DEFINER=`root`@`localhost`

TRIGGER мешканець\_оплатив\_комунальний\_платіж\_Before\_Delete BEFORE DELETE ON `мешканець оплатив комунальний платіж` FOR EACH ROW

BEGIN

insert into журнал\_операцій values( current\_user(), now(), 'видалення запису', old.`Номер квитанції`);

end$$

Результат роботи тригерів зображено на рисунку 4.



Рисунок 4 – Результат роботи тригерів

Висновки:

В результаті виконання лабораторної роботи були створені різноманітні тригери для бази даних, розробленої в попередніх лабораторних роботах. Робота тригерів була протестована на коректність

## Лабораторна робота № 6 Побудова ER-моделі предметної області.

Мета роботи: отримати навички з виконання первинних етапів проектування бази даних у сучасних середовищах.

Зміст звіту

1. Постановка задачі.
2. Програмний код
3. Копії екрану, що підтверджують виконання програми
4. Висновок

Завдання

1. Побудувати діаграму варіантів використання для програмного забезпечення та виконати її аналіз для виявлення тих функцій ПЗ, які потребують збереження, вилучення та обробки даних.
2. Виділити сутності предметної області та їх атрибути.
3. Встановити зв’язки між виділеними сутностями.
4. Назвати виділені зв’язки та встановити їх кратність.
5. Відобразити виділені сутності та зв’язки за допомогою обраної нотації.
6. Оформити звіт.

Варіант №6

Відділ продаж фірми

Програмне забезпечення повинне забезпечувати облік замовлень на поставку виготовлених фірмою виробів.

В програмі повинні бути передбачені наступні функції:

* облік виготовлених виробів та норм матеріалів для їх виготовлення;
* облік матеріалів;
* облік клієнтів фірми;
* облік замовлень на виготовлення виробів;
* формування квитанції на оплату замовлення;
* аналіз можливості реалізації замовлення з врахуванням наявних ресурсів;
* закупка необхідних матеріалів (формування закупівельної відомості);
* система знижок для постійних клієнтів.

**Хід виконання роботи**

На рисунку 1 представлена діаграма варіантів використання для системи, яка забезпечує облік замовлень на поставку виготовлених фірмою виробів.

В результаті проведеного аналізу діаграми варіантів використання були виділені такі варіанти використання, які потребують взаємодії з БД, а саме:

* додати данні клієнта;
* редагувати данні клієнта;
* видалити данні клієнта;
* розрахувати знижку;
* формування квитанції на оплату замовлення;
* формування замовлення;
* видалити замовлення;
* аналіз можливості реалізації замовлення;
* редагувати замовлення;
* додати виріб;
* видалити виріб;
* редагувати виріб;
* додати матеріали;
* видалити матеріали;
* редагувати матеріали;
* формування закупівельної відомості.



Рисунок 1 – Діаграма варіантів використання системи обліку замовлень на поставку виготовлених фірмою виробів

В результаті аналізу функцій розроблюваного ПЗ були виділені данні, які будуть зберігатись в БД: данні про клієнтів, замовлення, вироби та матеріали.

На основі даних, які будуть зберігатись в БД в системі обліку замовлень на поставку виготовлених фірмою виробів, буде відбуватись формування замовлень клієнтів, аналіз можливості реалізації цих замовлень з поточними ресурсами, а також формуватиметься закупівельна відомість та генеруватиметься квитанція на оплату замовлення.

В результаті аналізу предметної області були виділені сутності, що представлені на рисунку 2.



Рисунок 2 – Сутності предметної області

Після виділення сутностей предметної області необхідно визначити атрибути сутності. На рисунку 3 приведені виділенні сутності з атрибутами, серед яких є помічені як ключові атрибути.



Рисунок 3 – Атрибути та первинні ключі сутностей предметної області

Наступним етапом побудування ER-моделі є визначення зав’язків між сутностями. На рисунку 4 приведені зв’язки між виділеними сутностями.



Рисунок 4 – Зв’язки між сутностями предметної області

Всі виділенні зв’язки між сутностями є обов’язковими. Таким чином, на рисунку 5 приведена ER-модель для системи, яка забезпечує облік замовлень на поставку виготовлених фірмою виробів, побудована за допомогою нотації IDEF1x и CASE-засобу ERWin Data Modeler 7.0.



Рисунок 5 – ER-модель предметної області в нотації IDEF1x

Висновки:

В результаті виконання лабораторної роботи було побудовано діаграму варіантів використання та виділено сутності, атрибути та зв’язки для програмного забезпечення обліку замовлень на поставку виготовлених фірмою виробів.

## Лабораторна робота № 7 Побудова логічної моделі бази даних.

Мета роботи: отримати навички побудови логічної моделі даних шляхом нормалізації відношень у сучасних інструментальних середовищах

Зміст звіту

1. Постановка задачі.
2. Програмний код
3. Копії екрану, що підтверджують виконання програми
4. Висновок

Завдання

1. Визначити набір таблиць бази даних.
2. Реалізувати зв’язки між таблицями.
3. Перевірити отримані таблиці на відповідність першій, другій, третій нормальній формам та нормальній формі Бойса-Кодда.
4. В словесній формі сформувати запити до таблиць для покриття їх функціональності ПЗ.
5. В словесній формі сформувати призначення зберігаємих процедур та функцій, а також тригерів для забезпечення функціональності розроблюваного програмного забезпечення.

**Приклад виконання лабораторної роботи №7**

Варіант №6 – Відділ продаж фірми

На етапі логічного проектування БД необхідно на основі побудованої ER-моделі предметної області отримати схему реляційної бази даних.

Для побудування реляційної схеми БД необхідно визначити набір відношень (таблиць) БД.

Перетворення ER-моделі в схему реляційної БД за допомогою CASE-засобу ERWin Data Modeler 7.0 виконую за наступними правилами:

* кожній сутності ER-моделі ставиться у відповідність таблиця БД;
* зв’язки один-до-одного, один-до-багатьох реалізуються шляхом міграції первинного ключа батьківської сутності в дочірню.

На рисунку 1 представлена схема таблиць БД для забезпечення обліку замовлень на поставку виготовлених фірмою виробів.



Рисунок 1 – Схема БД

Описання всіх таблиць схеми реляційної ЮД для забезпечення обліку замовлень на поставку виготовлених фірмою виробів.

Таблиця «Клієнти»

|  |  |  |
| --- | --- | --- |
| *Атрибути* | *Тип* | *Описання* |
| іd\_Клієнта | Ціле число | Первинний ключ |
| Контактна\_особа | Рядок | Макс. довжина рядку = 100 |
| Номер\_телефону | Рядок | Макс. довжина рядку = 13 |
| Адреса | Рядок | Макс. довжина рядку = 100 |
| Знижка | Дійсне число | Формат числа Х.ХХ (числа>0) |

Таблиця «Замовлення»

|  |  |  |
| --- | --- | --- |
| *Атрибути* | *Тип* | *Описання* |
| №Замовлення | Ціле число | Первинний ключ |
| id\_Виробу | Ціле число | Зовнішній ключ |
| id\_Клієнта | Ціле число | Зовнішній ключ |
| Дата\_замовлення | Дата | Формат дати ГГГГ-ММ-ДД |
| Стан\_замовлення | Рядок | Приймає одне зі значень: «Виконано», «Очікуються матеріали» |
| Кількість\_виробів | Ціле число |  |
| До\_оплати | Дійсне число | Формат числа Х.ХХ |

Таблиця «Вироби»

|  |  |  |
| --- | --- | --- |
| *Атрибути* | *Тип* | *Описання* |
| іd\_Виробу | Ціле число | Первинний ключ |
| Назва\_виробу | Рядок | Макс. довжина рядку = 70 |
| Націнка,% | Дійсне число | Формат числа Х.ХХ (числа>0) |

Таблиця «Матеріали»

|  |  |  |
| --- | --- | --- |
| *Атрибути* | *Тип* | *Описання* |
| іd\_Матеріалу | Ціле число | Первинний ключ |
| Назва\_матеріалу | Рядок | Макс. довжина рядку = 70 |
| Одиниця\_виміру | Рядок | Макс. довжина рядку = 70 |

Таблиця «Матеріали\_на\_складі»

|  |  |  |
| --- | --- | --- |
| *Атрибути* | *Тип* | *Описання* |
| іd\_Матеріалу\_на\_складі | Ціле число | Первинний ключ |
| іd\_Матеріалу | Ціле число | Зовнішній ключ |
| Дата\_поставки | Дата | Формат дати ГГГГ-ММ-ДД |
| Ціна\_за\_одиницю | Дійсне число | Формат числа Х.ХХ |
| Кількість\_поставлених\_матеріалів | Дійсне число | Формат числа Х.ХХ |
| Залишилось\_на\_складі | Дійсне число | Формат числа Х.ХХ |

Перевіримо побудовані відношення на відповідність третій нормальній формі.

Наведені відношення знаходяться в першій нормальній формі (1NF), так як всі атрибути відношень приймають прості значення (отамарні або неділимі), які не є множинами або кортежем з більш елементарних складових.

Наведені відношення знаходяться в другій нормальній формі (2NF), так як вони знаходяться в першій нормальній формі, та кожний неключовий атрибут мінімально функціонально залежить від свого первинного ключа.

Наведені відношення знаходять в третій нормальній формі (3NF), так як вони знаходяться в другій нормальній формі, і кожний неключовий атрибут конкретного відношення нетранзитивно функціонально залежить від свого первинного ключа.

Наведені відношення знаходяться в нормальній формі Бойса-Кодда (BCNF), так як вони знаходяться в третій нормальній формі, так як в них відсутні залежності атрибутів первинного ключа від неключових атрибутів.

Запит на формування закупівельної відомості – який виводить загальну кількість та вартість поставлених матеріалів за датою поставки.

Запит, що виводить загальну суму замовлень та кількість всіх виробів.

Процедура, яка за id\_матеріала, його кількістю та ціною за одиницю оформлює поставку на склад за поточною датою.

Процедура, яка аналізує можливість реалізації замовлення, номер якого вказано в параметрі процедури, та реалізує його, якщо це можливо.

Функція Квитанція\_за\_номером – яка за номером замовлення виводить квитанцію на оплату цього замовлення.

Функція Ціна\_виробу – яка за id виробу розраховує його ціну, в яку входить ціна матеріалу помножена на кількість матеріалів, які необхідні для виготовлення цього виробу. Зі складу обирається матеріал з найменшою ціною.

Функція Аналіз\_замовлення, яка перевіряє наявність необхідних матеріалів для виконання замовлення та повертає одне зі значень «Очікуються матеріали» або «Виконано».

Тригер для таблиці Матеріали на складі, що спрацьовує при оформленні поставки матеріалу на склад та встановлює значення поля Залишилось\_на\_складі рівним полю Кількість\_поставлених\_матеріалів.

Тригер для таблиці Замовлення, що спрацьовує при формуванні замовлення та перевіряє наявність необхідних матеріалів для виконання замовлення та відображає результат в полі Стан\_замовлення та розраховує значення поля До\_оплати по формулі: (Вартість виробу \* Надбавка,% \* Знижка,%) \* Кількість\_виробів.

Процедура, яка розраховує знижку клієнтів в залежності від загальної суми замовлень за наступними правилами:

* знижка 2%, загальна сума замовлень від 3000 грн. до 4500 грн.;
* знижка 3%, загальна сума замовлень від 4500 грн. до 7000 грн.;
* знижка 5%, загальна сума замовлень від 7000 грн. до 10000 грн.;
* знижка 7%, загальна сума замовлень від 10000 грн. до 13000 грн.;
* знижка 8%, загальна сума замовлень від 13000 грн. до 15500 грн.;
* знижка 10%, загальна сума замовлень більше 15500 грн.

Висновки:

В результаті виконання лабораторної роботи було визначено набір таблиць БД та перевірено їх відповідність нормальним формам. Також в словесній формі було сформульовано запити до таблиць для покриття функціональності ПЗ, а також сформульовано призначення зберігаємих процедур, функцій та тригерів.

## Лабораторна робота № 8 Побудова фізичної моделі даних.

Мета роботи: отримати навички побудови фізичної моделі даних у сучасних інструментальних середовищах

Зміст звіту

1. Постановка задачі.
2. Програмний код
3. Копії екрану, що підтверджують виконання програми
4. Висновок

Завдання

1. Перетворити зв’язки «багато-до-багатьох» між таблицями (створити асоціативну таблицю з додатковими атрибутами, якщо це необхідно).
2. Обґрунтовано вибрати систему управління базами даних (СУБД).
3. Назначити стовбцям таблиць типи даних, які підтримує вибрана СУБД, встановити обмеження на значення стовбців таблиць бази даних.
4. Отримати SQL-скрипти для створення таблиць бази даних.
5. Створити таблиці бази даних (виконати SQL-скрипти, отримані в попередньому пункті).
6. Заповнити створені таблиці даними.

Приклад виконання лабораторної роботи

Варіант №6

Відділ продаж фірми

В першу чергу перетворюю зв’язок «багато-до-багатьох» шляхом створення асоціативної таблиці «Норма\_матеріалів». Після цього починаю розробку фізичної моделі.

Для системи, яка забезпечує облік замовлень на поставку виготовлених фірмою виробів було вибрано СУБД MySQL та інструмент для роботи з ним - MySQL Workbench. Я обрав цю СУБД через те, що: вона підтримується на основних платформах Microsoft Windows, Mac OS X и Linux; вона має можливість представлення моделі БД в графічному вигляді, а також редагування даних в таблиці; вона має простий та функціональний механізм створення зав’язків між полями таблиць; вона має редактор SQL-запитів, який дає можливість при відправлені на сервер отримувати відповідь в табличному вигляді та інші можливості.

В результаті призначення конкретних типів даних атрибутам таблиць було отримано наступну схему БД, яка представлена на рисунку 1.



Рисунок 1 – Схема БД з конкретними типами атрибутів

SQL-скрипти таблиць бази даних:

Клієнти:

CREATE TABLE `клієнти` (

 `id\_Клієнта` int(11) NOT NULL AUTO\_INCREMENT,

 `Контактна\_особа` varchar(100) NOT NULL,

 `Номер\_телефону` varchar(13) NOT NULL,

 `Адресса` varchar(100) NOT NULL,

 `Знижка,%` double(10,2) NOT NULL,

 PRIMARY KEY (`id\_Клієнта`)

) ENGINE=InnoDB AUTO\_INCREMENT=9 DEFAULT CHARSET=cp1251;

Вироби:

CREATE TABLE `вироби` (

 `id\_Виробу` int(11) NOT NULL AUTO\_INCREMENT,

 `Назва\_виробу` varchar(70) NOT NULL,

 `Націнка,%` double(10,2) NOT NULL,

 PRIMARY KEY (`id\_Виробу`)

) ENGINE=InnoDB AUTO\_INCREMENT=20 DEFAULT CHARSET=cp1251;

Матеріали:

CREATE TABLE `матеріали` (

 `id\_Матеріалу` int(11) NOT NULL AUTO\_INCREMENT,

 `Назва\_матеріалу` varchar(70) NOT NULL,

 `Одиниця\_виміру` varchar(70) NOT NULL,

 PRIMARY KEY (`id\_Матеріалу`)

) ENGINE=InnoDB AUTO\_INCREMENT=8 DEFAULT CHARSET=cp1251;

Замовлення:

CREATE TABLE `замовлення` (

 `№Замовлення` int(11) NOT NULL AUTO\_INCREMENT,

 `id\_Виробу` int(11) NOT NULL,

 `id\_Клієнта` int(11) NOT NULL,

 `Дата\_замовлення` date NOT NULL,

 `Кількість\_виробів` int(10) NOT NULL,

 `Стан\_замовлення` enum('Виконано','Очікуються матеріали') DEFAULT NULL,

 `До\_оплати` double(10,2) DEFAULT NULL,

 PRIMARY KEY (`№Замовлення`),

 KEY `id\_Виробу\_idx` (`id\_Виробу`),

 KEY `id\_Клієнта\_idx` (`id\_Клієнта`),

 CONSTRAINT `id\_Виробу` FOREIGN KEY (`id\_Виробу`) REFERENCES `вироби` (`id\_Виробу`) ON DELETE NO ACTION ON UPDATE NO ACTION,

 CONSTRAINT `id\_Клієнта` FOREIGN KEY (`id\_Клієнта`) REFERENCES `клієнти` (`id\_Клієнта`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=cp1251;

Норма\_матеріалів:

CREATE TABLE `норма\_матеріалів` (

 `id\_Виробу` int(11) NOT NULL,

 `id\_Матеріалу` int(11) NOT NULL,

 `Кількість` double(10,2) NOT NULL,

 PRIMARY KEY (`id\_Матеріалу`,`id\_Виробу`),

 KEY `id\_Виробу\_Норма\_матеріалів` (`id\_Виробу`),

 CONSTRAINT `id\_Виробу\_Норма\_матеріалів` FOREIGN KEY (`id\_Виробу`) REFERENCES `вироби` (`id\_Виробу`) ON DELETE NO ACTION ON UPDATE NO ACTION,

 CONSTRAINT `id\_Матеріалу\_Норма\_матеріалів` FOREIGN KEY (`id\_Матеріалу`) REFERENCES `матеріали` (`id\_Матеріалу`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=cp1251;

Матеріали\_на\_складі:

CREATE TABLE `матеріали\_на\_складі` (

 `id\_Матеріалу\_на\_складі` int(11) NOT NULL AUTO\_INCREMENT,

 `Дата\_поставки` date NOT NULL,

 `id\_Матеріалу` int(11) NOT NULL,

 `Кількість\_поставлених\_матеріалів` double(10,2) NOT NULL,

 `Ціна\_за\_одиницю` int(10) NOT NULL,

 `Залишилось\_на\_складі` double(10,2) DEFAULT NULL,

 PRIMARY KEY (`id\_Матеріалу\_на\_складі`),

 KEY `id\_Матеріалу\_idx` (`id\_Матеріалу`),

 CONSTRAINT `id\_Матеріалу` FOREIGN KEY (`id\_Матеріалу`) REFERENCES `матеріали` (`id\_Матеріалу`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB AUTO\_INCREMENT=8 DEFAULT CHARSET=cp1251;

Створив таблиці та заповнив їх даними. Для перегляду записів були створені наступні запити:

SELECT \* FROM відділ\_продаж\_фірми.вироби;

Результат зображено на рисунку 2.



Рисунок 2 – Записи таблиці «Вироби»

 SELECT \* FROM відділ\_продаж\_фірми.клієнти;

Результат зображено на рисунку 3.



Рисунок 3 – Записи таблиці «Клієнти»

SELECT \* FROM відділ\_продаж\_фірми.матеріали;

Результат зображено на рисунку 4.



Рисунок 4 – Записи таблиці «Матеріали»

SELECT \* FROM відділ\_продаж\_фірми.матеріали\_на\_складі;

Результат зображено на рисунку 5.



Рисунок 5 – Записи таблиці «Матеріали\_на\_складі»

SELECT \* FROM відділ\_продаж\_фірми.норма\_матеріалів;

Результат зображено на рисунку 6.



Рисунок 6 – Записи таблиці «Норма\_матеріалів»

Таблицю «Замовлення» буде заповнено під час наступної лабораторної роботи, так як на даний момент не реалізовані тригери для цієї таблиці.

**Висновки:**

В результаті виконання лабораторної роботи логічну модель бази даних було перетворено в фізично та реалізовано її в СУБД MySQL за допомогою інструмента MySQl Workbench

## Лабораторна робота № 9 Розробка SQL-скриптів запитів, зберігаємих процедур та функцій, тригерів.

Мета роботи: отримання навичок розроблення скриптів до існуючих баз даних

Зміст звіту

1. Постановка задачі.
2. Програмний код
3. Копії екрану, що підтверджують виконання програми
4. Висновок

Завдання

1. Скласти, реалізувати та протестувати SQL-скрипти запитів для забезпечення функціональності розроблюваного програмного забезпечення.
2. Скласти, реалізувати та протестувати SQL-скрипти зберігаємих процедур та функцій для забезпечення функціональності розроблюваного програмного забезпечення.
3. Скласти, реалізувати та протестувати SQL-скрипти тригерів для забезпечення функціональності розроблюваного програмного забезпечення.

Варіант №6

Відділ продаж фірми

Процедура, яка за id\_матеріала, його кількістю та ціною за одиницю оформлює поставку на склад за поточною датою. Код процедури:

CREATE DEFINER=`root`@`localhost` PROCEDURE `Поставка\_на\_склад`(Матеріал int, Кількість double, Ціна int)

BEGIN

 INSERT INTO `відділ\_продаж\_фірми`.`матеріали\_на\_складі`

 (`id\_Матеріалу\_на\_складі`,

 `Дата\_поставки`,

 `id\_Матеріалу`,

 `Кількість\_поставлених\_матеріалів`,

 `Ціна\_за\_одиницю`,

 `Залишилось\_на\_складі`)

 VALUES

 (null, now(), Матеріал, Кількість, Ціна, Кількість);

END

Результат роботи процедури зображено на рисунку 1 для вхідних даних:

call відділ\_продаж\_фірми.Поставка\_на\_склад(4, 3, 280).



Рисунок 1– Результат роботи процедури

Функція Аналіз\_замовлення, яка перевіряє наявність необхідних матеріалів для виконання замовлення та повертає одне зі значень «Очікуються матеріали» або «Виконано». Код функції:

CREATE DEFINER=`root`@`localhost` FUNCTION `Аналіз\_замовлення`(Виріб int, Кількість int) RETURNS varchar(200) CHARSET cp1251

BEGIN

 declare no\_more\_rows1 int default 0;

 declare Матеріал int(11);

 declare Кількість\_на\_складі double(10,2);

 declare Необхідна\_кількість double(10,2);

 declare result varchar(200);

 declare Необхідні\_матеріали Cursor for Select id\_Матеріалу, норма\_матеріалів.Кількість from норма\_матеріалів where id\_Виробу=Виріб;

 DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET no\_more\_rows1 =1;

 open Необхідні\_матеріали;

 set result='Виконано';

 while no\_more\_rows1 =0 do

 fetch Необхідні\_матеріали into Матеріал, Необхідна\_кількість;

 if no\_more\_rows1 =0 then

 set Необхідна\_кількість=Необхідна\_кількість\*Кількість;

 SELECT SUM(Залишилось\_на\_складі) INTO Кількість\_на\_складі FROM матеріали\_на\_складі

 WHERE id\_Матеріалу = Матеріал GROUP BY id\_Матеріалу;

 if (Необхідна\_кількість>Кількість\_на\_складі)

 then

 set result='Очікуються матеріали';

 end if;

 end if;

 end while;

 close Необхідні\_матеріали;

RETURN result;

END

Результат роботи функції зображено на рисунку 2:



Рисунок 2– Результат роботи функції

Функція Ціна\_виробу – яка за id виробу розраховує його ціну, в яку входить ціна матеріалу помножена на кількість матеріалів, які необхідні для виготовлення цього виробу. Зі складу обирається матеріал з найменшою ціною. Код функції:

CREATE DEFINER=`root`@`localhost` FUNCTION `Ціна\_Виробу`(Виріб int) RETURNS varchar(200) CHARSET cp1251

BEGIN

 declare main\_result varchar(200);

 declare result varchar(200);

 declare no\_more\_rows1 int default 0;

 declare no\_more\_rows2 int default 0;

 declare Кількість\_на\_складі double(10,2);

 declare Необхідна\_кількість double(10,2);

 declare Матеріал int(11);

 declare Варість\_виробу double(10,2) default 0;

 declare Вартість\_матеріалів double(10,2);

 declare Вартість\_матеріалу double(10,2);

 declare Кількість\_матеріалу double(10,2);

 declare Націнка double(10,2);

 declare До\_оплати double(10,2);

 declare Стан\_замовлення varchar(20);

 declare Кількість\_виробів double(10,2);

 declare Необхідні\_матеріали Cursor for Select id\_Матеріалу, Кількість from норма\_матеріалів where id\_Виробу=Виріб;

 DECLARE CONTINUE HANDLER FOR not found SET no\_more\_rows1 =1;

 set result=Аналіз\_замовлення(Виріб);

 if (result='Виконано')

 then

 set Варість\_виробу=0;

 open Необхідні\_матеріали;

 while no\_more\_rows1 =0 do

 fetch Необхідні\_матеріали into Матеріал, Необхідна\_кількість;

 if no\_more\_rows1 =0 then

 set Вартість\_матеріалів=0;

 BLOCK2: begin

 declare Доступні\_матеріали cursor for

 select Залишилось\_на\_складі, Ціна\_за\_одиницю from матеріали\_на\_складі

 where id\_Матеріалу=Матеріал order by Ціна\_за\_одиницю;

 DECLARE CONTINUE HANDLER for not found SET no\_more\_rows2 =1;

 SET no\_more\_rows2 = 0;

 set Кількість\_на\_складі=0;

 set Вартість\_матеріалів=0;

 open Доступні\_матеріали;

 while no\_more\_rows2 =0 do

 fetch Доступні\_матеріали into Кількість\_матеріалу, Вартість\_матеріалу;

 if no\_more\_rows2 =0 then

 set Кількість\_на\_складі=Кількість\_на\_складі+Кількість\_матеріалу;

 if (Кількість\_матеріалу<Необхідна\_кількість) then

 set Вартість\_матеріалів=Вартість\_матеріалів+(Вартість\_матеріалу\*Кількість\_матеріалу);

 set Необхідна\_кількість=Необхідна\_кількість-Кількість\_матеріалу;

 else

 if (Кількість\_матеріалу=Необхідна\_кількість) then

 set Вартість\_матеріалів=Вартість\_матеріалів+(Вартість\_матеріалу\*Кількість\_матеріалу);

 set Необхідна\_кількість=Необхідна\_кількість-Кількість\_матеріалу;

 set no\_more\_rows2=1;

 else if (Кількість\_матеріалу>Необхідна\_кількість) then

 set Вартість\_матеріалів=Вартість\_матеріалів+(Вартість\_матеріалу\*Необхідна\_кількість);

 set no\_more\_rows2=1;

 end if;

 end if;

 end if;

 end if;

 End while;

 close Доступні\_матеріали;

 select `Націнка,%` into Націнка from вироби where id\_Виробу=Виріб;

 set Варість\_виробу=Варість\_виробу+Вартість\_матеріалів;

 end BLOCK2;

 end if;

 End while;

 close Необхідні\_матеріали;

 set До\_оплати=Варість\_виробу \* (1+Націнка/100);

 select Назва\_виробу into main\_result from вироби where id\_Виробу=Виріб;

 set result=Concat('Ціна виробу "',main\_result,'" - ',До\_оплати,' грн.');

 end if;

 if (result='Очікуються матеріали') then

 set main\_result='На складі недостатньо матеріалів для розрахунку ціни!';

 else

 set main\_result=result;

 end if;

RETURN main\_result;

END

Результат роботи функції зображено на рисунку 3:



Рисунок 3– Результат роботи функції

Тригер для таблиці Матеріали на складі, що спрацьовує при оформленні поставки матеріалу на склад та встановлює значення поля Залишилось\_на\_складі рівним полю Кількість\_поставлених\_матеріалів. Код тригеру:

CREATE DEFINER=`root`@`localhost` TRIGGER `відділ\_продаж\_фірми`.`матеріали\_на\_складі\_BEFORE\_INSERT` BEFORE INSERT ON `матеріали\_на\_складі` FOR EACH ROW

BEGIN

 set new.Залишилось\_на\_складі=new.Кількість\_поставлених\_матеріалів;

END

Процедура, яка розраховує знижку клієнту в залежності від загальної суми замовлень за наступними правилами:

* знижка 2%, загальна сума замовлень від 3000 грн. до 4500 грн.;
* знижка 3%, загальна сума замовлень від 4500 грн. до 7000 грн.;
* знижка 5%, загальна сума замовлень від 7000 грн. до 10000 грн.;
* знижка 7%, загальна сума замовлень від 10000 грн. до 13000 грн.;
* знижка 8%, загальна сума замовлень від 13000 грн. до 15500 грн.;
* знижка 10%, загальна сума замовлень більше 15500 грн.

Код процедури:

CREATE DEFINER=`root`@`localhost` PROCEDURE `Оновити\_знижку`(Код\_Клієнта int)

BEGIN

 declare Загальна\_сума double(10,2);

 declare Процент double(10,2);

 select sum(До\_оплати) into Загальна\_сума from замовлення where id\_Клієнта=Код\_Клієнта group by id\_Клієнта;

 if (Загальна\_сума>=3000) and (Загальна\_сума<4500)

 then

 set Процент=2;

 else

 if (Загальна\_сума>=4500) and (Загальна\_сума<7000)

 then

 set Процент=3;

 else

 if (Загальна\_сума>=7000) and (Загальна\_сума<10000)

 then

 set Процент=5;

 else

 if (Загальна\_сума>=10000) and (Загальна\_сума<13000)

 then

 set Процент=7;

 else

 if (Загальна\_сума>=13000) and (Загальна\_сума<15500)

 then

 set Процент=8;

 else

 if (Загальна\_сума>=15500)

 then

 set Процент=10;

 end if;

 end if;

 end if;

 end if;

 end if;

 end if;

 if (Процент>0)

 then

 UPDATE `відділ\_продаж\_фірми`.`клієнти` SET `Знижка,%` = Процент

 WHERE `id\_Клієнта` = Код\_Клієнта;

 end if;

END

Тригер для таблиці Замовлення, що спрацьовує при формуванні замовлення та перевіряє наявність необхідних матеріалів для виконання замовлення та відображає результат в полі Стан\_замовлення та розраховує значення поля До\_оплати по формулі: (Вартість виробу \* Надбавка,% \* Знижка,%) \* Кількість\_виробів.

Код тригеру:

CREATE DEFINER=`root`@`localhost` trigger Перевірка\_можливості\_реалізаці\_BI

before insert on замовлення

for each row begin

BLOCK1: begin

 declare no\_more\_rows1 int default 0;

 declare no\_more\_rows2 int default 0;

 declare inc int default 0;

 declare Кількість\_на\_складі double(10,2);

 declare Необхідна\_кількість double(10,2);

 declare Матеріал int(11);

 declare Варість\_виробу double(10,2) default 0;

 declare Вартість\_матеріалів double(10,2);

 declare Вартість\_матеріалу double(10,2);

 declare Кількість\_матеріалу double(10,2);

 declare id\_Матеріалу\_зі\_складу int(11);

 declare Націнка double(10,2);

 declare Знижка double(10,2);

 declare result varchar(200);

 declare Необхідні\_матеріали Cursor for Select id\_Матеріалу, Кількість from норма\_матеріалів where id\_Виробу=new.id\_Виробу;

 DECLARE CONTINUE HANDLER FOR not found SET no\_more\_rows1 =1;

 set result=Аналіз\_замовлення(new.id\_Виробу, new.Кількість\_виробів);

 if (result='Виконано')

 then

 set Варість\_виробу=0;

 open Необхідні\_матеріали;

 while no\_more\_rows1 =0 do

 fetch Необхідні\_матеріали into Матеріал, Необхідна\_кількість;

 if no\_more\_rows1 =0 then

 set Необхідна\_кількість=Необхідна\_кількість\*new.Кількість\_виробів;

 set Вартість\_матеріалів=0;

 BLOCK2: begin

 declare Доступні\_матеріали cursor for

 select id\_Матеріалу\_на\_складі, Залишилось\_на\_складі, Ціна\_за\_одиницю from матеріали\_на\_складі

 where id\_Матеріалу=Матеріал order by Ціна\_за\_одиницю;

 DECLARE CONTINUE HANDLER for not found SET no\_more\_rows2 =1;

 SET no\_more\_rows2 = 0;

 set Кількість\_на\_складі=0;

 set Вартість\_матеріалів=0;

 open Доступні\_матеріали;

 while no\_more\_rows2 =0 do

 fetch Доступні\_матеріали into id\_Матеріалу\_зі\_складу, Кількість\_матеріалу, Вартість\_матеріалу;

 if no\_more\_rows2 =0 then

 set Кількість\_на\_складі=Кількість\_на\_складі+Кількість\_матеріалу;

 if (Кількість\_матеріалу<Необхідна\_кількість) then

 set Вартість\_матеріалів=Вартість\_матеріалів+(Вартість\_матеріалу\*Кількість\_матеріалу); UPDATE `відділ\_продаж\_фірми`.`матеріали\_на\_складі` SET `Залишилось\_на\_складі` = 0

 WHERE `id\_Матеріалу\_на\_складі` = id\_Матеріалу\_зі\_складу;

 set Необхідна\_кількість=Необхідна\_кількість-Кількість\_матеріалу;

 else

 if (Кількість\_матеріалу=Необхідна\_кількість) then

 set Вартість\_матеріалів=Вартість\_матеріалів+(Вартість\_матеріалу\*Кількість\_матеріалу); UPDATE `відділ\_продаж\_фірми`.`матеріали\_на\_складі` SET `Залишилось\_на\_складі` = 0

 WHERE `id\_Матеріалу\_на\_складі` = id\_Матеріалу\_зі\_складу;

 set Необхідна\_кількість=Необхідна\_кількість-Кількість\_матеріалу;

 set no\_more\_rows2=1;

 else if (Кількість\_матеріалу>Необхідна\_кількість) then

 set Вартість\_матеріалів=Вартість\_матеріалів+(Вартість\_матеріалу\*Необхідна\_кількість);

 UPDATE `відділ\_продаж\_фірми`.`матеріали\_на\_складі` SET `Залишилось\_на\_складі` = Кількість\_матеріалу-Необхідна\_кількість

 WHERE `id\_Матеріалу\_на\_складі` = id\_Матеріалу\_зі\_складу;

 set no\_more\_rows2=1;

 end if;

 end if;

 end if;

 end if;

 End while;

 close Доступні\_матеріали;

 select `Націнка,%` into Націнка from вироби where id\_Виробу=new.id\_Виробу;

 select `Знижка,%` into Знижка from клієнти where id\_Клієнта=new.id\_Клієнта;

 set Варість\_виробу=Варість\_виробу+Вартість\_матеріалів ;

 end BLOCK2;

 end if;

 End while;

 close Необхідні\_матеріали;

 set new.До\_оплати=Варість\_виробу \* (1+Націнка/100) \* (1- Знижка/100);

 end if;

 select №Замовлення into inc from замовлення order by №Замовлення desc limit 1;

 set new.№Замовлення=inc+1;

 set new.Стан\_замовлення=result;

end BLOCK1;

end

Для перевірки тригеру виконаємо наступний запит:

INSERT INTO `відділ\_продаж\_фірми`.`замовлення` (`id\_Виробу`, `id\_Клієнта`, `Дата\_замовлення`, `Кількість\_виробів`) VALUES ('13', '2', '2019-12-08', '1');

Результат запиту зображено на рисунку 4.



Рисунок 4 - Результат запиту

Матеріали, які необхідні для виготовлення виробу із запиту вище зображено на рисунку 5.



Рисунок 5 – Записи таблиці «Норма\_матеріалів»

Кількість матеріалів на складі до та після виконання замовлення зображено на рисунках 6, 7.



Рисунок 6 – Записи таблиці «Матеріали\_на\_складі»



Рисунок 7 – Записи таблиці «Матеріали\_на\_складі»

Процедура, яка аналізує можливість реалізації замовлення, номер якого вказано в параметрі процедури, та реалізує його, якщо це можливо.

Код процедури:

CREATE DEFINER=`root`@`localhost` PROCEDURE `Виконати\_замовлення`(Замовлення int(11))

BEGIN

BLOCK1: begin

 declare no\_more\_rows1 int default 0;

 declare no\_more\_rows2 int default 0;

 declare inc int default 0;

 declare Кількість\_на\_складі double(10,2);

 declare Необхідна\_кількість double(10,2);

 declare Матеріал int(11);

 declare Варість\_виробу double(10,2) default 0;

 declare Вартість\_матеріалів double(10,2);

 declare Вартість\_матеріалу double(10,2);

 declare Кількість\_матеріалу double(10,2);

 declare id\_Матеріалу\_зі\_складу int(11);

 declare Націнка double(10,2);

 declare Знижка double(10,2);

 declare result varchar(200);

 declare Виріб int(11);

 declare Клієнт int(11);

 declare Стан\_замовлення varchar(20);

 declare Кількість\_виробів double(10,2);

 declare Необхідні\_матеріали Cursor for Select id\_Матеріалу, Кількість from норма\_матеріалів where id\_Виробу=(Select id\_Виробу from замовлення where №Замовлення=Замовлення);

 DECLARE CONTINUE HANDLER FOR not found SET no\_more\_rows1 =1;

 select id\_Виробу, id\_Клієнта, замовлення.Кількість\_виробів, замовлення.Стан\_замовлення into Виріб, Клієнт, Кількість\_виробів, Стан\_замовлення from замовлення where №Замовлення=Замовлення;

 if (Стан\_замовлення<>'Виконано') then

 set result=Аналіз\_замовлення(Виріб, Кількість\_виробів);

 if (result='Виконано')

 then

 set Варість\_виробу=0;

 open Необхідні\_матеріали;

 while no\_more\_rows1 =0 do

 fetch Необхідні\_матеріали into Матеріал, Необхідна\_кількість;

 if no\_more\_rows1 =0 then

 set Необхідна\_кількість=Необхідна\_кількість\*Кількість\_виробів;

 set Вартість\_матеріалів=0;

 BLOCK2: begin

 declare Доступні\_матеріали cursor for

 select id\_Матеріалу\_на\_складі, Залишилось\_на\_складі, Ціна\_за\_одиницю from матеріали\_на\_складі

 where id\_Матеріалу=Матеріал order by Ціна\_за\_одиницю;

 DECLARE CONTINUE HANDLER for not found SET no\_more\_rows2 =1;

 SET no\_more\_rows2 = 0;

 set Кількість\_на\_складі=0;

 set Вартість\_матеріалів=0;

 open Доступні\_матеріали;

 while no\_more\_rows2 =0 do

 fetch Доступні\_матеріали into id\_Матеріалу\_зі\_складу, Кількість\_матеріалу, Вартість\_матеріалу;

 if no\_more\_rows2 =0 then

 set Кількість\_на\_складі=Кількість\_на\_складі+Кількість\_матеріалу;

 if (Кількість\_матеріалу<Необхідна\_кількість) then

 set Вартість\_матеріалів=Вартість\_матеріалів+(Вартість\_матеріалу\*Кількість\_матеріалу); UPDATE `відділ\_продаж\_фірми`.`матеріали\_на\_складі` SET `Залишилось\_на\_складі` = 0

 WHERE `id\_Матеріалу\_на\_складі` = id\_Матеріалу\_зі\_складу;

 set Необхідна\_кількість=Необхідна\_кількість-Кількість\_матеріалу;

 else

 if (Кількість\_матеріалу=Необхідна\_кількість) then

 set Вартість\_матеріалів=Вартість\_матеріалів+(Вартість\_матеріалу\*Кількість\_матеріалу); UPDATE `відділ\_продаж\_фірми`.`матеріали\_на\_складі` SET `Залишилось\_на\_складі` = 0

 WHERE `id\_Матеріалу\_на\_складі` = id\_Матеріалу\_зі\_складу;

 set Необхідна\_кількість=Необхідна\_кількість-Кількість\_матеріалу;

 set no\_more\_rows2=1;

 else if (Кількість\_матеріалу>Необхідна\_кількість) then

 set Вартість\_матеріалів=Вартість\_матеріалів+(Вартість\_матеріалу\*Необхідна\_кількість);

 UPDATE `відділ\_продаж\_фірми`.`матеріали\_на\_складі` SET `Залишилось\_на\_складі` = Кількість\_матеріалу-Необхідна\_кількість

 WHERE `id\_Матеріалу\_на\_складі` = id\_Матеріалу\_зі\_складу;

 set no\_more\_rows2=1;

 end if;

 end if;

 end if;

 end if;

 End while;

 close Доступні\_матеріали;

 select `Націнка,%` into Націнка from вироби where id\_Виробу=Виріб;

 set Варість\_виробу=Варість\_виробу+Вартість\_матеріалів;

 end BLOCK2;

 end if;

 End while;

 close Необхідні\_матеріали;

 select `Знижка,%` into Знижка from клієнти where id\_Клієнта=Клієнт;

 update замовлення set До\_оплати=Варість\_виробу \* (1+Націнка/100) \* (1- Знижка/100) where №Замовлення=Замовлення;

 end if;

 update замовлення set Стан\_замовлення=result where №Замовлення=Замовлення;

 end if;

end BLOCK1;

END

Функція Квитанція\_за\_номером – яка за номером замовлення виводить квитанцію на оплату цього замовлення.

Код функції:

CREATE DEFINER=`root`@`localhost` FUNCTION `Квитанція\_для\_замовлення`(№Замовлення int) RETURNS varchar(200) CHARSET cp1251

BEGIN

 declare result varchar(200);

 declare Виріб, Клієнт, Стан varchar(100);

 declare Дата date;

 declare Кількість, Оплата double;

 select Назва\_виробу, Контактна\_особа, Дата\_замовлення, Кількість\_виробів, До\_оплати, замовлення.Стан\_замовлення

 into Виріб, Клієнт, Дата, Кількість, Оплата, Стан

 from клієнти, вироби, замовлення

 where (клієнти.id\_Клієнта=замовлення.id\_Клієнта) and (вироби.id\_Виробу=замовлення.id\_Виробу) and (замовлення.№Замовлення=№Замовлення);

 if (Стан='Виконано') then

 set result=Concat('Квитанцію видано: ',Дата,'. Замовлено "',Виріб,'" у кількості: ',Кількість,'. Оплачує особа: ',Клієнт,'. Всього до оплати: ',Оплата,' грн.');

 else

 set result='Замовлення ще не виконано';

 end if;

RETURN result;

END

Результат роботи функції зображено на рисунку 8:



Рисунок 8 – Результат роботи функції

Запит на формування закупівельної відомості – який виводить загальну кількість та вартість поставлених матеріалів за датою поставки.

Код запиту:

select Назва\_матеріалу as Матеріал, Дата\_поставки, Кількість\_поставлених\_матеріалів, Ціна\_за\_одиницю, Кількість\_поставлених\_матеріалів\*Ціна\_за\_одиницю as Вартість

from матеріали join матеріали\_на\_складі on матеріали.id\_Матеріалу=матеріали\_на\_складі.id\_Матеріалу

Результат роботи запиту зображено на рисунку 9.



Рисунок 9 – Результат роботи запиту

Запит, що виводить загальну суму замовлень та кількість всіх виробів.

Код запиту:

select Назва\_виробу, sum(До\_оплати) as Загальна\_сума, sum(Кількість\_виробів) as Загальна\_кількість

from вироби left join замовлення on вироби.id\_Виробу=замовлення.id\_Виробу

group by вироби.id\_Виробу

 Результат роботи запиту зображено на рисунку 10.



Рисунок 10 – Результат роботи запиту

Висновки:

В результаті виконання лабораторної роботи були створені різноманітні тригери для бази даних, процедури та функції для забезпечення функціональності програмного забезпечення. Робота тригерів була протестована на коректність. Також були проведені їх виклики та перевірки на працездатність.

# Тематика індивідуальних завдань

Не передбачено навчальним планом

# Методичні вказівки до самостійної роботи студентів

Самостійна робота студента (СРС) - це форма організації навчального процесу, при якій заплановані завдання виконуються студентом під методичним керівництвом викладача, але без його безпосередньої участі. СРС с основним засобом засвоєння навчального матеріалу під час поза аудиторної навчальної роботи. СРС спрямована на закріплення теоретичних знань, отриманих студентами за час навчання, їх поглиблений, набуття і удосконалення практичних навичок та умінь відповідно до обраного напряму підготовки. Відтак, метою самостійної роботи студентів є набуття додаткових знань, перевірка отриманих знань на практиці, вироблення фахових та дослідницьких вмінь та навичок.

Самостійна робота студентів включає:

− підготовку до аудиторних занять (лекцій, практичних);

− виконання завдань з навчальної дисципліни протягом семестру;

− роботу над окремими темами навчальних дисциплін, які згідно з робочою навчальною програмою дисципліни винесені на самостійне опрацювання студентів;

− підготовку до підсумкових контрольних випробувань.

Самостійна робота студента над засвоєнням навчального матеріалу дисципліни може виконуватися у бібліотеці, навчальних кабінетах, а також в домашніх умовах. Самостійна робота студента забезпечується системою навчально-методичних засобів, передбачених для вивчення конкретної навчальної дисципліни. Зміст самостійної роботи студента над конкретною дисципліною визначають навчальна програма, завдання та вказівки викладача. Таким чином, самостійна робота є організованою викладачем активною діяльністю студента, направленою на виконання поставленої дидактичної мети. За своєю суттю самостійна робота є активною розумовою діяльністю студента, пов’язаною з виконанням навчального завдання. Під час самостійної роботи студенти мають змогу краще використати свої індивідуальні здібності. Вони вивчають, конспектують літературні джерела, за потреби повторно перечитують, звертаються до відповідних довідників і словників.

Самостійна робота здійснює і виховний вплив на студентів, сприяючи формуванню і розвитку необхідних моральних якостей.

Розподіл годин для самостійної роботи

|  |  |  |
| --- | --- | --- |
| №з/п | Назва теми | Кількість годин |
| 1 | Реляційна модель даних. | 3 |
| 2 | Проектування бази даних.  | 3 |
| 3 | Відношення та їхні властивості | 3 |
| 4 | Мова DDL. | 3 |
| 5 | Мова DML. | 4 |
| 6 | Складні запити даних | 4 |
| 7 | Елементи об'єктно-орієнтованого програмування у базах даних. | 4 |
| 8 | Процедури MySQL. | 4 |
| 9 | Курсори у базах даних | 4 |
| 10 | Тригери у базах даних. | 4 |
| 11 | Захист даних | 4 |
| 12 | Розроблення розосереджених баз даних | 4 |
| Разом | 44 |

# Питання, тести для контрольних заходів

## Змістовий модуль 1:

* 1. Основні задачі БД.
	2. Етапи проектування БД.
	3. Моделі даних, застосовні на різних етапах.
	4. Реляційна модель даних.
	5. Концептуальна, логічна, фізична моделі даних.
	6. Основні елементи моделей даних.
	7. Різниці між моделями даних.
	8. Сутності та їх атрибути.
	9. Моделювання сутностей відношеннями.
	10. Моделювання відношень таблицями.
	11. Елементи відношень.
	12. Елементи таблиць.
	13. Мова створення елементів баз даних.
	14. Типи даних.
	15. Послідовності (лічильники).
	16. Створення бази даних, таблиць, стовпчиків та зв’язків.
	17. Відтворення, внесення, модифікація та видалення даних з таблиць бази даних
	18. Проектування бази даних.
	19. Визначення вимог до системи.
	20. Збір і аналізування вимог користувачів.
	21. Проектування бази даних.
	22. Нормалізація даних як шлях проектування БД.
	23. Перша, друга та треті нормальні форми.
	24. Четвнерта нормальна форма та її властивості.
	25. Пята нормальна форма та її використання.
	26. Приведеня моделі даних до шостої нормпальної форми.
	27. Вибір цільової системи керування базою даних.
	28. Властивості систем керування базами даних.
	29. Розроблення додатків.
	30. Створення прототипів
	31. Конвертування і завантаження даних.
	32. Тестування бази даних.
	33. Експлуатація і супровід баз даних.
	34. Відношення та їх властивості.
	35. ER-метод проектування.
	36. Трансформація ER-моделі в реляційну модель даних.
	37. Обмеження цілісності реляційної моделі даних.
	38. Види цілісності моделей даних.
	39. Підтримка цілісності баз даних.
	40. Підтримка цілісності сутностей та зв’язків.
	41. Денормалізація даних та рефакторинг бази даних для підвищення її продуктивності.
	42. Шифрування даних.
	43. Контрзаходи і некомп’ютерні засоби контролю доступу до даних.
	44. Підходи до забезпечення безпеки і планування захисту від непередбачених обставин.

## Змістовий модуль 2:

* 1. Мова DDL та її основи.
	2. Основні типи даних.
	3. Реалізація даних в MySQL.
	4. Мова визначення даних DDL SQL.
	5. Схема даних.
	6. Типи даних, що підтримує система керування БД.
	7. Таблиці, стовпчики таблиць, подання, обмеження, ключ, зв’язки та особливості їх створення мовою DDL.
	8. Кардинальність та призначення зв’язків.
	9. Рольові зв’язки.
	10. Звязки між різними таблицями.
	11. Рекурсивні зв’язки.
	12. Звязки тип-підтип.
	13. Звязки типу 1:1, 1:К, К:М та їх використання у моделях даних.
	14. Мова маніпулювання даними DML SQL
	15. Відтворення даних.
	16. Перевірка умов.
	17. Запити до даних.
	18. Команда добування даних SELECT.
	19. Однотабличні запити, запити з умовами, запити на об'єднання даних.
	20. Корельвані запити.
	21. Запити з об’єднанням таблиць.
	22. Використання аліасів стовпчиків та таблиць у запитах до баз даних.
	23. Складні запити до бази даних.
	24. Багатотабличні запити, кореляційні запити, запити з агрегацією.
	25. Агрегація з умовою.
	26. Агрегатні функції та їх використання для проектування баз даних.
	27. Агрегація з аласами.
	28. Способи підвищення продуктивності агрегованих запитів.
	29. Об'єктні типи у МySQL.
	30. Розміщення об'єктів у базі даних.
	31. Використання об'єктних типів.
	32. Геоінформаційні типи даних.
	33. Створення та використання об’єктних типів під час проектування бази даних.
	34. Запити з об’єктними типами даних.

## Змістовий модуль 3:

* 1. Процедури у MySQL.
	2. Структура блока PL/SQL.
	3. Оголошення змінних у PL/SQL.
	4. Типи даних у мові PL/SQL.
	5. Скалярні і складені типи (записи та таблиці).
	6. Використання SQL у коді PL/SQL.
	7. Вбудовані функції PL/SQL.
	8. Процедури та функції мовою PL/SQL та їх написання.
	9. Завантаження процедур у БД.
	10. Виконання процедур та функцій. Параметри в процедурах та функціях PL/SQL.
	11. Курсори у БД як засіб пришвидшення доступу до баз даних.
	12. Типи курсорів.
	13. Реалізація курсорів у процедурах PL/SQL.
	14. Особливості синтаксису курсорів.
	15. Способи використання курсорів для збірання певних даних у процедурах та функціях.
	16. Тригери у БД.
	17. Призначення тригерів.
	18. Створення тригерів.
	19. Типи тригерів.
	20. Тригери для операція вставки, модифікації та видаленя записів з таблиць.
	21. Тригери до та після виконання операцій з модифікації записів.
	22. Використання функцій та процедур у тригерах.
	23. Способи оптимізації тригерів.
	24. Реалізація каскадного відновлення даних у реляційних базах даних

## Змістовий модуль 4:

* 1. Захист даних у БД
	2. Транзакції, фіксація та відкат.
	3. Цілісність даних.
	4. Моделі транзакцій.
	5. Відновлення бази даних.
	6. Поліпшені моделі транзакцій.
	7. Модель вкладених транзакцій.
	8. Керування транзакціями.
	9. Підтримка транзакцій.
	10. Створення транзакцій мовою TCL.
	11. Реалізація транзакцій мовами TCL та PL/SQL.
	12. Відкат транзакцій. Штатне завершення транзакцій.
	13. Розроблення розосереджених баз даних.
	14. Архітектури баз даних.
	15. Файл-серверна архітектура БД
	16. Клієнт-серверна архітектура БД
	17. Трирівнева архітектура БД.
	18. Розподілені бази даних та їх використання.
	19. Концепції і розроблення розподілених БД.
	20. Принципи організації комп'ютерних мереж.
	21. Функції й архітектура розподілених БД.
	22. Керування доступом до бБД.
	23. Привілеї, користувачі і ролі.
	24. Мова DCL як засіб контролю привілеями та користувачами.
	25. Визначення привілеїв, ролей та користувачів у базах даних.

# Питання для самоперевірки

* 1. Реляційна модель даних.
	2. Концептуальна, логічна та фізична моделі даних.
	3. Основні елементи моделей даних.
	4. Сутності та їх атрибути.
	5. Елементи відношень та таблиць.
	6. Типи даних БД.
	7. Створення бази даних, таблиць, стовпчиків та зв’язків.
	8. Нормалізація даних як шлях проектування БД..
	9. Обмеження цілісності реляційної моделі даних.
	10. Денормалізація даних та ре факторинг бази даних для підвищення її продуктивності.
	11. Мова визначення даних DDL SQL.
	12. Кардинальність та призначення зв’язків
	13. Команди маніпулювання даними DML SQL.
	14. Команда добування даних SELECT.
	15. Однотабличні запити, запити з умовами, запити на об'єднання даних.
	16. Корельвані запити.
	17. Запити з об’єднанням таблиць.
	18. Складні запити до бази даних.
	19. Багатотабличні запити, кореляційні запити, запити з агрегацією.
	20. Агрегатні функції та їх використання для проектування баз даних.
	21. Об'єктні типи у МySQL.
	22. Геоінформаційні типи даних.
	23. Створення та використання об’єктних типів під час проектування бази даних.
	24. Запити з об’єктними типами даних.
	25. Процедури та функції у MySQL.
	26. Скалярні і складені типи (записи та таблиці).
	27. Використання SQL у коді PL/SQL.
	28. Вбудовані функції PL/SQL.
	29. Процедури та функції мовою PL/SQL та їх написання.
	30. Завантаження процедур у БД
	31. Виконання процедур та функцій.
	32. Параметри в процедурах та функціях.
	33. Типи курсорів.
	34. Реалізація курсорів у процедурах PL/SQL.
	35. Особливості синтаксису курсорів..
	36. Тригери у БД.
	37. Призначення тригерів.
	38. Тригери для операція вставки, модифікації та видаленя записів з таблиць.
	39. Використання функцій та процедур у тригерах.
	40. Способи оптимізації тригерів
	41. Транзакції, фіксація та відкат.
	42. Цілісність даних.
	43. Моделі транзакцій.
	44. Відновлення бази даних.
	45. Поліпшені моделі транзакцій.
	46. Модель вкладених транзакцій.
	47. Керування транзакціями.
	48. Створення транзакцій мовою TCL.
	49. Відкат транзакцій.
	50. Штатне завершення транзакцій.
	51. Архітектури баз даних.
	52. Файл-серверна архітектура БД
	53. Клієнт-серверна архітектура БД
	54. Триривнева архітектура БД.
	55. Розподілені бази даних та їх використання.
	56. Концепції і розроблення розподілених СУБД.
	57. Функції й архітектура розподілених СУБД.
	58. Керування доступом до баз даних.
	59. Привілеї, користувачі і ролі.
	60. Мова DCL як засіб контролю привілеями та користувачами.

# Рекомендована література, інформаційні ресурси

## Базова

1. Дунаєв, В.В. Бази даних. Мова SQL для студента. - 2-е вид., Доп. і перераб. [Текст] / В.В. Дунаєв - СПб .: БХВ-Петербург, 2007. URL: http://znanium.com/bookread.php?book=350372
2. Малихіна, М.П. Бази даних: основи, проектування, використання: навчальний посібник [Текст] / М. П. Малихіна, Санкт-Петербург: БХВ-Петербург, 2004, 512 с.
3. Мартишин, С.А. Проектування і реалізація баз даних в СУБД MySQL з використанням MySQL Workbench: Навчальний посібник [Текст] / С.А. Мартишин і ін. - М .: ИД ФОРУМ: НДЦ Инфра-М, 2012. - 160 с. URL: http://www.znanium.com/bookread.php?book=318518

## Допоміжна

1. Пирогов, В. Ю. Інформаційні системи і бази даних: організація та проектування: навч. посібник [Текст] / В. Ю. Пирогов. СПб .: БХВ-Петербург, 2009. - 528 с. URL: http: //znanium.com/bookread.php? Book = 350672
2. Туманов, В.Є. Проектування сховищ даних для систем бізнес-аналітики: навчальний посібник [Текст] / В.Е.Туманов, Москва: Інтернет-Університет Інформаційних Технологій: БИНОМ. Лабораторія знань, 2011, 615 с.

## Інформаційні ресурси

1. Документація для програмістів, що розробляють БД MY SQL https://dev.mysql.com/doc/refman/8.0/en/creating-database.html
2. Сервер з інформацією про застосування БД http://www.java2s.com
3. Сервер з інформацією про застосування БД у середовищі Інтернет https://www.w3schools.com
4. Докуметація з БД MY SQL http://www.mysqltutorial.org/mysql-create-database