МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ПРИКАРПАТСЬКИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИлЯ СТЕФАНИКА

ФАКУЛЬТЕТ МАТЕМАТИКИ І ІНФоРМАТИКИ

КАФЕДРА КОМп’ЮТЕРНИХ НАУК

**проектування та розробка бази даних**

З НАВЧАЛЬНОГО ПРЕДМЕТА: “База даних”

**ТЕмА**

**Магазин комп’ютерної техніки**

Виконали:

студенти групи КН-21

Павлюк М.М.,

Ровінський Ю.В.

……………………….

(особистий підпис, дата)

Прийняла:

к.т.н., Семаньків М.В.

……………………….

(особистий підпис, дата)

м. Івано-Франківськ

2021 р.

**Хід роботи.**

Спочатку було проаналізовано тему ”Магазин комп’ютерної техніки” та визначено наступні сутності, які повинні бути в проектованій базі даних:

* замовлення (чеки);
* продукти (товари);
* працівники;
* покупці.

У результаті було отримано початкову ER-діаграму (рис. 1). Початкову, а також кінцеву діаграму, можна переглянути за [даним](https://drive.google.com/file/d/12EhidmCKeefwBjqc7CAwBw7z0ZN5nqNd/view?usp=sharing) посиланням.

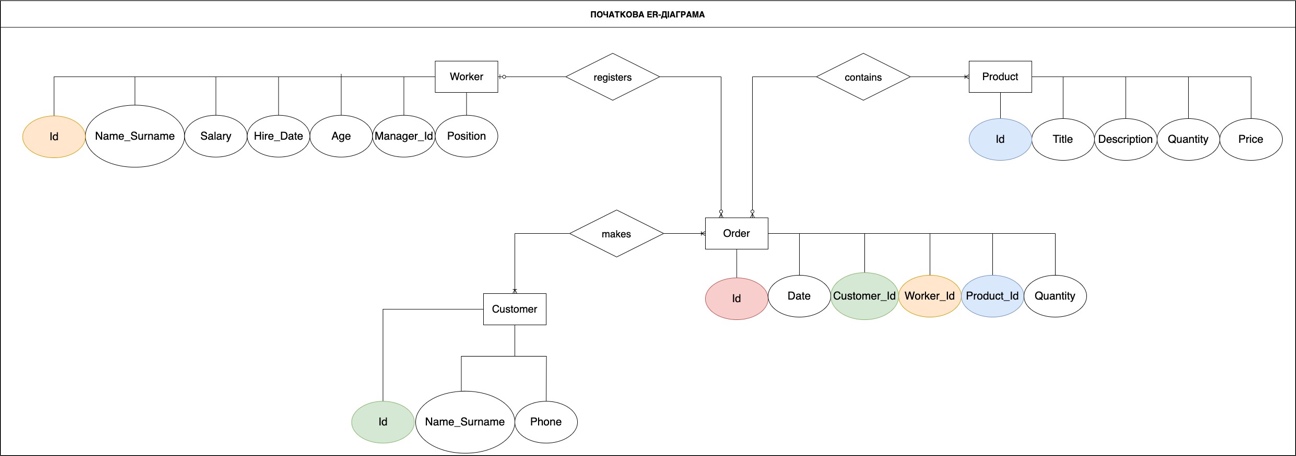


Рисунок 1 – Початкова ER-діаграма

Розглянемо атрибути таблиць:

Worker:

* Id;
* Name\_Surname – ім’я та прізвище;
* Salary – зарплата;
* Hire\_Date – дата найму;
* Age – вік;
* Manager\_Id – id працівника, який є менеджером для даного воркера;
* Position – назва посади.

Product:

* Id;
* Title – назва;
* Description – опис;
* Quantity – наявна кількість;
* Price – ціна за одиницю продукту.

Customer:

* Id;
* Name\_Surname – ім’я та прізвище;
* Phone – номер телефону в міжнародному форматі.

Order:

* Id;
* Date – дата пробиття чеку;
* Customer\_Id – id покупця;
* Worker\_Id – id працівника, який оформив чек;
* Product\_Id – id товару;
* Quantity – кількість товару.

Так як в нас зв’язок між Order та Product – багато-до-багатьох, то для того, щоб реалізувати даний зв’язок необхідно створити третю проміжну зв’язуючу таблицю. В даній таблиці будуть міститись первинні ключі двох таблиць. Також за рахунок створення проміжної таблиці (Order\_Detail) в нас не будуть дублюватись: дата, працівник та покупець, якщо в чеку буде декілька товарів. Тобто за рахунок розділення чеку на основну (Order) та підлеглу (Order\_Detail) таблиці ми також здійснюємо нормалізацію нашої бази даних.

На етапі нормалізації було виявлено, що атрибут Name\_Surname в таблицях Worker та Customer не є атомарним, тому для того, щоб задовільнити умови першої нормальної форми, було вирішено розділити даний атрибут на два атомарних: Name та Surname. Також після більш детального аналізу таблиць на предмет перебування в другій та третій нормальних формах було виявлено те, що таблиця Worker містить атрибут Salary, який не залежить від первинного ключа. Так як зарплата залежить від посади, то щоб друга нормальна форма була досягнута, було вирішено створити таблицю Position, яка б містила назву посади та зарплату, а в Worker ми б мали тільки Position\_Id.

Після реалізації наявних зв’язків та процесу нормалізації було отримано кінцеву ER-діаграму бази даних (рис. 2).

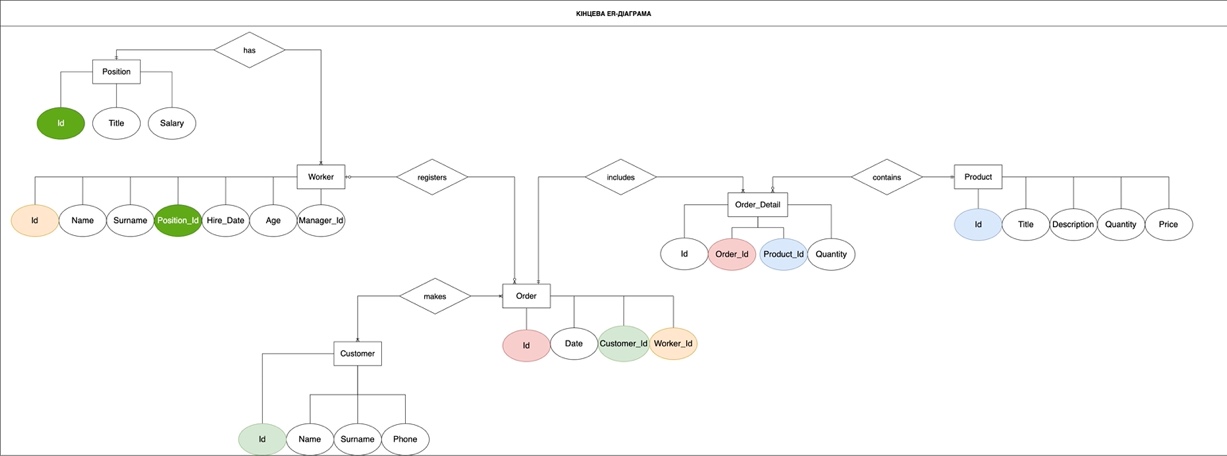


Рисунок 2 – Кінцева ER-діаграма

Так як на діаграмах одразу були позначені різними кольорами залежні Id, то логічну реляційну модель (одразу нормалізовану) у вигляді таблиць з первинними та зовнішніми ключами було зображено таким чином (рис. 3):

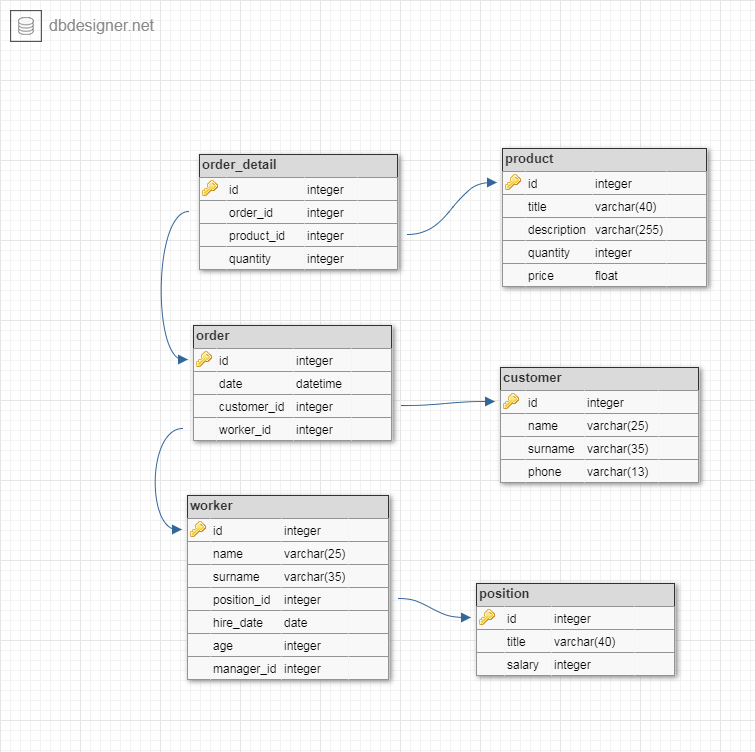


Рисунок 3 ‒ Логічна реляційна модель

Для фізичного представлення даної бази було обрано MySQL СУБД. Запити для створення всіх таблиць:

**CREATE TABLE rp.`order` (**

**`id` INT NOT NULL AUTO\_INCREMENT,**

**`date` DATETIME NOT NULL,**

**`customer\_id` INT NOT NULL,**

**`worker\_id` INT NOT NULL,**

**PRIMARY KEY (`id`)**

**);**

**CREATE TABLE rp.`customer` (**

**`id` INT NOT NULL AUTO\_INCREMENT,**

**`name` VARCHAR(25) NOT NULL,**

**`surname` VARCHAR(35) NOT NULL,**

**`phone` VARCHAR(13) NOT NULL,**

**PRIMARY KEY (`id`)**

**);**

**CREATE TABLE rp.`worker` (**

**`id` INT NOT NULL AUTO\_INCREMENT,**

**`name` VARCHAR(25) NOT NULL,**

**`surname` VARCHAR(35) NOT NULL,**

**`position\_id` INT NOT NULL,**

**`hire\_date` DATE NOT NULL,**

**`age` INT NOT NULL,**

**`manager\_id` INT NOT NULL,**

**PRIMARY KEY (`id`)**

**);**

**CREATE TABLE rp.`position` (**

**`id` INT NOT NULL AUTO\_INCREMENT,**

**`title` VARCHAR(40) NOT NULL,**

**`salary` INT NOT NULL,**

**PRIMARY KEY (`id`)**

**);**

**CREATE TABLE rp.`product` (**

**`id` INT NOT NULL AUTO\_INCREMENT,**

**`title` VARCHAR(40) NOT NULL,**

**`description` VARCHAR(255) NOT NULL,**

**`quantity` INT NOT NULL,**

**`price` FLOAT NOT NULL,**

**PRIMARY KEY (`id`)**

**);**

**CREATE TABLE rp.`order\_detail` (**

**`id` INT NOT NULL AUTO\_INCREMENT,**

**`order\_id` INT NOT NULL,**

**`product\_id` INT NOT NULL,**

**`quantity` INT NOT NULL,**

**PRIMARY KEY (`id`)**

**);**

**ALTER TABLE rp.`order` ADD CONSTRAINT `order\_fk0` FOREIGN KEY (`customer\_id`) REFERENCES `customer` (`id`);**

**ALTER TABLE rp.`order` ADD CONSTRAINT `order\_fk1` FOREIGN KEY (`worker\_id`) REFERENCES `worker` (`id`);**

**ALTER TABLE rp.`worker` ADD CONSTRAINT `worker\_fk0` FOREIGN KEY (`position\_id`) REFERENCES `position` (`id`);**

**ALTER TABLE rp.`order\_detail` ADD CONSTRAINT `order\_detail\_fk0` FOREIGN KEY (`order\_id`) REFERENCES `order` (`id`);**

**ALTER TABLE rp.`order\_detail` ADD CONSTRAINT `order\_detail\_fk1` FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`);**

Лістинг 1 – Код створення спроектованої бази даних

Також для того, щоб мати можливість видалення та оновлення даних зв’язаних таблиць (задля цілісності даних), було використано REFERENCES.

**ALTER TABLE rp.order\_detail DROP FOREIGN KEY order\_detail\_fk1;**

**ALTER TABLE rp.order\_detail ADD CONSTRAINT order\_detail\_fk1 FOREIGN KEY (product\_id) REFERENCES rp.product (id) ON DELETE CASCADE ON UPDATE CASCADE;**

**ALTER TABLE rp.order\_detail DROP FOREIGN KEY order\_detail\_fk0;**

**ALTER TABLE rp.order\_detail ADD CONSTRAINT order\_detail\_fk0 FOREIGN KEY (order\_id) REFERENCES rp.order (id) ON DELETE CASCADE ON UPDATE CASCADE;**

**ALTER TABLE rp.order DROP FOREIGN KEY order\_fk0, DROP FOREIGN KEY order\_fk1;**

**ALTER TABLE rp.order**

**ADD CONSTRAINT order\_fk0 FOREIGN KEY (customer\_id) REFERENCES rp.customer(id) ON UPDATE CASCADE,**

**ADD CONSTRAINT order\_fk1 FOREIGN KEY (worker\_id) REFERENCES rp.worker(id) ON UPDATE CASCADE;**

**ALTER TABLE rp.worker DROP FOREIGN KEY worker\_fk0;**

**ALTER TABLE rp.worker ADD CONSTRAINT worker\_fk0 FOREIGN KEY (position\_id) REFERENCES rp.position (id) ON UPDATE CASCADE;**

Лістинг 2 – Код для забезпечення цілісності даних

**Запити:**

1. Видалити товар з id = 3.

**DELETE**

**FROM rp.product**

**WHERE id = 3;**

1. Видалити чек, id якого дорівнює 2.

**DELETE**

**FROM rp.order**

**WHERE id = 2;**

1. Змінити id чеку з id = 6 на 7.

**UPDATE rp.order**

**SET id = 7**

**WHERE id = 6;**

1. Змінити id посади з id = 1 на 10.

**UPDATE rp.position**

**SET id = 10**

**WHERE id = 1;**

1. Вивести унікальні імена працівників.

**SELECT DISTINCT name**

**FROM rp.worker;**

1. Вивести інформацію про СЕО магазину (СЕО нікому не підпорядкується – порожнє поле manager\_id)

**SELECT \***

**FROM rp.worker**

**WHERE manager\_id IS NULL;**

1. Виведіть інформацію про три найдешевші продукти.

**SELECT \***

**FROM rp.product**

**ORDER BY price limit 3;**

1. Хто з працівників найдовше працює?

**SELECT name,**

**surname**

**FROM rp.worker**

**WHERE hire\_date = (**

**SELECT min(hire\_date)**

**FROM rp.worker**

**);**

1. Вивести продукт, якого найбільше в наявності і його кількість.

**SELECT title,**

**quantity**

**FROM rp.product**

**WHERE quantity = (**

**SELECT max(quantity)**

**FROM rp.product**

**);**

1. Виведіть середню зарплату з усіх посад.

**SELECT avg(salary)**

**FROM rp.position;**

1. Яку кількість чеків було пробито в період між 1 березня 2021 і 31 березня 2021? Назву стовпця замініть на “OrderQuantity”.

**SELECT count(id) AS OrderQuantity**

**FROM rp.order**

**WHERE date BETWEEN '2021-03-01'**

**AND '2021-03-31';**

1. Вивести загальну кількість усіх наявних товарів. Назву стовпця замініть на “TotalQuantity”.

**SELECT sum(quantity) AS TotalQuantity**

**FROM rp.product;**

1. Вивести всі чеки, які було сформовано починаючи від 12 години дня.

**SELECT \***

**FROM rp.order**

**WHERE hour(date) >= 12;**

1. Вивести всі чеки, які було оброблено цього року.

**SELECT \***

**FROM rp.order**

**WHERE year(date) = year(CURRENT\_DATE ());**

1. Виведіть імена покупців, в яких ім’я починається на букву “А”.

**SELECT name**

**FROM rp.customer**

**WHERE name LIKE 'A%';**

1. Вивести інформацію про продукт, назва якого починається з “IPhone”.

**SELECT \***

**FROM rp.product**

**WHERE title LIKE "IPhone%";**

1. Виведіть можливий заробіток по кожному виду товару з розрахунком на його наявну кількість та на одиничну ціну. Стовпчик нехай має назву “TotalProfit”.

**SELECT (price \* quantity) AS TotalProfit**

**FROM rp.product;**

1. Відсортуйте продукти за спаданням ціни.

**SELECT \***

**FROM rp.product**

**ORDER BY price DESC;**

1. Скільки кожного виду товару було продано?

**SELECT sum(quantity)**

**FROM rp.order\_detail**

**GROUP BY product\_id;**

1. Скільки кожного віку працівників працюють в магазині? Виведіть вік та кількість працівників даного віку. Стовпчик назвіть “WorkersQuantity”.

**SELECT age,**

**count(id) AS WorkersQuantity**

**FROM rp.worker**

**GROUP BY age;**

1. Яка кількість працівників влаштувалась на роботу у 2017 році?

**SELECT count(id)**

**FROM rp.worker**

**WHERE year(hire\_date) = 2017;**

1. Скільки замовлень було зроблено кожного місяця (при цьому місяці, в яких не було покупок - відкидаємо) впродовж 2021 року?

**SELECT month(date),**

**count(id)**

**FROM rp.order**

**WHERE year(date) = 2021**

**GROUP BY month(date);**

1. Скільки у кожного менеджера є підлеглих?

**SELECT manager\_id,**

**count(id)**

**FROM rp.worker**

**WHERE manager\_id IS NOT NULL**

**GROUP BY manager\_id;**

1. Вивести ціни продуктів, які зустрічаються в двох або більше продуктах.

**SELECT price**

**FROM rp.product**

**GROUP BY price**

**HAVING count(id) >= 2;**

1. Погрупуйте наявні номера покупців за кодом країни абонента (перші три символи) та виведіть їх кількість.

**SELECT substr(phone, 1, 3) AS CountryCode,**

**count(id)**

**FROM rp.customer**

**GROUP BY substr(phone, 1, 3);**

1. Вивести працівника та його менеджера (4 колонки – прізвище та ім’я спочатку працівника, а потім прізвище та ім’я його менеджера).

**SELECT a.name,**

**a.surname,**

**b.name,**

**b.surname**

**FROM rp.worker a,**

**rp.worker b**

**WHERE a.manager\_id = b.id;**

1. Вивести ім’я, прізвище та назву посади для кожного працівника.

**SELECT name,**

**surname,**

**title**

**FROM rp.worker AS a**

**JOIN rp.position AS b ON a.position\_id = b.id;**

1. Вивести товар, половину або більше половини кількості якого вже продали (залишилось на складі <=, ніж продали).

**SELECT DISTINCT a.title**

**FROM rp.product AS a**

**JOIN rp.order\_detail AS b ON a.id = b.product\_id**

**WHERE a.quantity <= b.quantity;**

1. Виведіть дату призначення “СЕО” магазину та його зарплату.

**SELECT hire\_date,**

**salary**

**FROM rp.worker AS a**

**JOIN rp.position AS b ON a.position\_id = b.id**

**WHERE b.title = 'CEO';**

1. Виведіть id чеків, які пробивали менеджери.

**SELECT a.id**

**FROM rp.order AS a**

**JOIN rp.worker AS b ON a.worker\_id = b.id**

**JOIN rp.position AS c ON b.position\_id = c.id**

**WHERE c.title = 'manager';**

1. Виведіть ім’я та прізвище працівників, які в день свого призначення на посаду пробили свій перший чек.

**SELECT name,**

**surname**

**FROM rp.worker AS a**

**JOIN rp.order AS b ON a.id = b.worker\_id**

**WHERE hire\_date = DATE (b.date);**

1. Вивести назву продукту, який ще не продавали.

**SELECT a.title**

**FROM rp.product AS a**

**LEFT JOIN rp.order\_detail AS b ON a.id = b.product\_id**

**WHERE b.product\_id IS NULL;**

1. Вивести id та дати чеків, які пробив наймолодший працівник.

**SELECT a.id,**

**a.date**

**FROM rp.order AS a**

**JOIN rp.worker AS b ON a.worker\_id = b.id**

**WHERE b.age = (**

**SELECT min(age)**

**FROM rp.worker**

**);**

1. Вивести id та дати чеків, які пробив працівник, який найменше працює в магазині.

**SELECT a.id,**

**a.date**

**FROM rp.order AS a**

**JOIN rp.worker AS b ON a.worker\_id = b.id**

**WHERE b.hire\_date = (**

**SELECT max(hire\_date)**

**FROM rp.worker**

**);**

1. Вивести всі чеки першого покупця.

**SELECT a.\***

**FROM rp.order AS a**

**JOIN rp.customer AS b ON a.customer\_id = b.id**

**WHERE b.id = (**

**SELECT min(id)**

**FROM rp.customer**

**);**

1. Вивести чеки, у яких покупець є абонентами Vodafone (+38050, +38099);

**SELECT a.\***

**FROM rp.order AS a**

**JOIN rp.customer AS b ON a.customer\_id = b.id**

**WHERE substr(b.phone, 1, 6) IN (**

**'+38050',**

**'+38099'**

**);**

1. Вивести повний чек (разом з id Product та кількістю товару).

**SELECT a.id,**

**a.date,**

**a.customer\_id,**

**a.worker\_id,**

**b.product\_id,**

**b.quantity**

**FROM rp.order AS a**

**JOIN rp.order\_detail AS b ON a.id = b.order\_id;**

1. Вивести чеки, які містять два або більше товарів.

**SELECT a.\***

**FROM rp.order AS a**

**JOIN rp.order\_detail AS b ON a.id = b.order\_id**

**GROUP BY a.id**

**HAVING count(b.id) >= 2;**

1. Вивести прізвище покупця та прізвище людини, яка оформляла чек.

**SELECT a.surname,**

**c.surname**

**FROM rp.customer AS a**

**JOIN rp.order AS b ON a.id = b.customer\_id**

**JOIN rp.worker AS c ON b.worker\_id = c.id;**

1. Вивести кожного покупця та загальну суму, яку він витратив у магазині

**SELECT a.id,**

**a.name,**

**a.surname,**

**sum(c.quantity \* d.price) AS Total**

**FROM rp.customer AS a**

**JOIN rp.order AS b ON a.id = b.customer\_id**

**JOIN rp.order\_detail AS c ON b.id = c.order\_id**

**JOIN rp.product AS d ON c.product\_id = d.id**

**GROUP BY a.id;**

1. Вивести кількість покупців, які купили найдорожчий товар.

**SELECT count(DISTINCT a.id) AS Majory**

**FROM rp.customer AS a**

**JOIN rp.order AS b ON a.id = b.customer\_id**

**JOIN rp.order\_detail AS c ON b.id = c.order\_id**

**JOIN rp.product AS d ON c.product\_id = d.id**

**WHERE d.price = (**

**SELECT max(price)**

**FROM rp.product**

**);**

1. Вивести працівників, які продали кожен тип наявних товарів.

**SELECT a.id,**

**a.name,**

**a.surname**

**FROM rp.worker AS a**

**JOIN rp.order AS b ON a.id = b.worker\_id**

**JOIN rp.order\_detail AS c ON b.id = c.order\_id**

**JOIN rp.product AS d ON c.product\_id = d.id**

**GROUP BY a.id**

**HAVING count(DISTINCT c.product\_id) = (**

**SELECT count(id)**

**FROM rp.product**

**);**

1. Вивести для кожного покупця його знижку (якщо загальна сума, яку він витратив в магазині більше 10000, то він отримає 3% знижку, якщо від 20000 до 30000 – 5%, більше 30000 – 7%).

**SELECT a.id,**

**a.name,**

**a.surname,**

**CASE**

**WHEN sum(c.quantity \* d.price) >= 10000**

**AND sum(c.quantity \* d.price) < 20000**

**THEN 3**

**WHEN sum(c.quantity \* d.price) >= 20000**

**AND sum(c.quantity \* d.price) < 30000**

**THEN 5**

**WHEN sum(c.quantity \* d.price) >= 30000**

**THEN 7**

**ELSE 0**

**END AS DiscountPercentage**

**FROM rp.customer AS a**

**JOIN rp.order AS b ON a.id = b.customer\_id**

**JOIN rp.order\_detail AS c ON b.id = c.order\_id**

**JOIN rp.product AS d ON c.product\_id = d.id**

**GROUP BY a.id;**

**Висновок**. Для локального тестування бази даних таблиці було заповнено даними. Всі запити дали очікуваний результат, тобто спроектована база даних відповідає всім вимогам та є зручною для користування за рахунок правильної реалізації зв’язків та виконання нормалізації.