DOI: 10.36910/6775-2524-0560-2020-39-01
УДК: 005
Булатецька Леся Віталіївна, канд. фіз.-мат. наук, доцент
https://orcid.org/0000-0002-7202-826X
Булатецький Віталій Вікторович канд. фіз.-мат. наук, доцент
https://orcid.org/0000-0002-9883-4550
Східноєвропейський національний університет імені Лесі Українки, м. Луцьк, Україна

ОСОБЛИВОСТІ ВИВЧЕННЯ МОВИ ЗАПИТІВ SQL В ПРОФІЛЬНОМУ КУРСІ ІНФОРМАТИКИ ЗАКЛАДІВ ЗАГАЛЬНОЇ СЕРЕДНЬОЇ ОСВІТИ

Л.В. Булатецька, В.В. Булатецький. Особливості вивчення мови запитів SQL в профільному курсі інформатики закладів загальної середньої освіти. Матеріал статті містить роз'яснення і методичні рекомендації для вивчення теми "мова запитів SQL" в профільному курсі інформатики закладів загальної середньої освіти. В роботі детально описано поняття цілісності даних в реляційній базі даних, проаналізовано ризики порушення цілісності даних при додаванні, оновленні та видаленні записів в багатотабличній базі даних та принципи побудови і порядок виконання запитів SQL на вибірку даних.
Ключові слова: методика вивчення, реляційні бази даних, мова запитів SQL, цілісність бази даних, між-табличні зв'язки.

Л.В. Булатецькая, В.В. Булатецкий. Особенности изучения языка запросов SQL в профильном курсе информатики заведений общего среднего образования. Материал статьи содержит разъяснения и методические рекомендации к изучению темы "язык запросов SQL" в профильном курсе информатики заведений общего среднего образования. В работе подробно описано понятие целостности данных в реляционной базе данных, проанализированы риски нарушения целостности данных при добавлении, обновлении и удалении записей в многотабличной базе данных, принципы построения и порядок выполнения запросов SQL на выборку данных.
Ключевые слова: методика изучения, реляционные базы данных, язык запросов SQL, целостность базы данных, меж-табличные связи.

L.V. Bulatetska, V.V. Bulatetsky. Features of studying the language of SQL queries in the profile course of informatics of institutions of general secondary education. The material of the article contains explanations and methodological recommendations for the study of the topic "SQL query language" in the profile course of informatics of institutions of general secondary education. The paper describes in detail the concept of data integrity in a relational database, analyzes the risks of data integrity breach when adding, updating, and deleting records in a multi-table database, and the principles for constructing and performing SQL queries for fetching data.
Keywords: study methodology, relational databases, SQL query language, database integrity, inter-table relationships.

Постановка проблеми та аналіз досліджень. Зміст навчальної програми профільного рівня та рівня стандарту вивчення інформатики для здобувачів 10-11 класів закладів загальної середньої освіти, містить розділ "Бази даних", який вивчається в 11 класі [1,2]. Вивчення даної теми проходить за підручником Завадського І. "Основи баз даних" [3]. В даному підручнику автор основну увагу приділяє проектуванню реляційних баз даних. Знайомство з реляційними базами даних відбувається на прикладі системи керування базами даних (СКБД) Access. У розділах розширеної версії даного підручника пропонується вивчення мови запитів SQL (Structured Query Language). Мову SQL вважають декларативною мовою, на відміну від процедурних мов, на яких пишуться програми. Це означає, що вирази на мові SQL описують, що потрібно зробити, а не яким чином. Після вивчення цієї теми, здобувач повинен знати та розуміти конструкції мови запитів, створювати та виконувати запити на вибірку даних з однієї та кількох зв'язаних таблиць, групувати дані, виконувати запити на введення даних в таблицю, їх зміну і видалення [1]. Автори робіт [3, 4], аналізуючи методичні особливості вивчення розділу "Бази даних" в закладах загальної середньої освіти, виділяють складності у розумінні окремих понять та теоретичних положень. Основні труднощі, які виникають при побудові запитів SQL у здобувачів загальної середньої освіти – це розуміння поняття обмеження цілісності відношень в реляційних базах даних та принципів організації зв'язків між таблицями [4].

Метою роботи є дослідження особливостей методики вивчення мови запитів SQL теми "Бази даних" в профільному курсі інформатики закладів загальної середньої освіти.

Виклад основного матеріалу й обґрунтування отриманих результатів дослідження. Для того, щоб розуміти написання конструкції запитів на вибірку та зміну даних, здобувачі загальної середньої освіти, повинні розуміти окремі поняття теорії реляційних баз даних. Реляційна модель даних була винайдена британським ученим Едгаром Франком Коддом, який вперше використав для моделювання даних математичні принципи теорії множин і математичної логіки. Щоб розуміти суть

завдання створення реляційної бази даних, а також операцій над даними, достатньо розглянути на теоретичному (абстрактному) рівні всього лише кілька основних положень теорії відношень. Таблиці, з яких складається будь-яка реляційна база даних, являють собою деякі відношення, а відношення є не чим іншим, як множинами. У термінології реляційних баз даних рядки таблиці називаються записами (кортежами), стовпці – полями (атрибутами), множину всіх можливих значень в стовпці – доменами. Всі запити до бази даних, спрямовані на вилучення з неї потрібних записів, інтерпретуються як інструкції з виконання тих чи інших операцій, що, в результаті є операціями алгебри множин [5].

Будь-яке відношення реляційної моделі даних має наступні основні характеристики:

- в кожному рядку таблиці мають міститися дані, які відповідають деякому об'єкту або його частині;
- в кожному стовпці повинні знаходитися дані, що відповідають одному з атрибутів відношення;
- в кожній клітинці таблиці повинне знаходитися тільки єдине значення;
- у кожного стовпця повинно бути унікальне ім'я;
- всі рядки (записи) у таблиці повинні бути різними;
- порядок розташування стовпців і рядків у таблиці не має значення [5].

На практиці доводиться проектувати базу даних як набір декількох таблиць, які спочатку порожні і тільки з часом заповнюються конкретними даними. Однак кожна таблиця окремо, а також сукупність таблиць, зазвичай є не випадковими комбінаціями атрибутів, довільно розподіленими між різними таблицями. Як окремі таблиці, так і вся їх сукупність, що називаються базою даних, володіють деякою цілісністю, яка виражається через різного роду обмеження, що накладаються на значення стовпців і зв'язки між ними.

Здобувачі загальної середньої освіти повинні чітко розуміти поняття цілісності даних в реляційній базі даних. Якщо таблиця в базі даних повністю відповідає деякому об'єкту реального світу, то говорять, що вона володіє семантичною цілісністю і моделює (представляє) цей об'єкт. Наприклад в базі даних "Школа" – таблиця "Учні"(з полями "Код_учня", "Прізвище", "Ім'я", "Дата народження", "Стать") моделює об'єкт реального світу "учень", тобто володіє семантичною цілісністю (Рис.1.). У таблиці, що має семантичну цілісність, повинен бути первинний ключ (PRIMARY KEY), який відіграє важливу роль при створенні таблиць реляційної бази даних. Первинний ключ – це множина з одного або декількох атрибутів, яка однозначно визначає (ідентифікує) весь запис у відношенні. Значення первинного ключа повинні бути унікальними і визначеними.

В таблиці "Учні" в якості первинного ключа взято поле "Код_учня". Первинний ключ задається при створенні таблиці, або після її створення засобами модифікації таблиці. Те, що поле "Код_учня" є первинним ключем, означає, що всі дані в даному полі є визначеними і такими, що не повторюються.

Учні

Код_учня	Прізвище	Ім'я	Дата_народження	Стать
1	Шпак	Максим	11.12.2009	ч
2	Григорчук	Петро	05.05.2009	ч
3	Хоменко	Ірина	07.10.2007	ж

Рис. 1. Вміст таблиці "Учні" бази даних "Школа"

При додаванні даних в таку таблицю, потрібно слідкувати, щоб не порушувалось обмеження цілісності первинного ключа. Наприклад, наступні два запити на додавання записів в таблицю "Учні" за допомогою команди INSERT, будуть виконані, оскільки значення стовпця, що визначає первинний ключ є визначеними і відрізняються один від одного.

INSERT INTO Учні (Код_учня, Прізвище, Ім'я, Дата_народження, стать) VALUES (4, 'Семенюк', 'Петро', #05.02.200#);

INSERT INTO Учні (Код_учня, Прізвище, Ім'я, Дата_народження, стать) VALUES (5, 'Семенюк', 'Петро', #05.02.2005#);

Якщо ми спробуємо виконати наступний запит на додавання даних:

INSERT INTO Учні (Код_учня, Прізвище, Ім'я, Дата_народження, стать) VALUES (3, 'Семенюк', 'Петро', #05.02.2005#);

то цей запит не виконається, так як порушується умова визначення первинного ключа. Всі значення стовпця "Код_учня" повинні бути такими, що відрізняються, а ми пробуємо добавити рядок зі значенням поля "Код_учня"=3, який вже існує в нашій таблиці.

Також при оновленні даних, ми не зможемо змінити значення поля "Код_учня" на значення, яке вже існує в базі даних. Тобто наступний запит на оновлення даних не виконається.

UPDATE Учні SET Код_учня = 3 WHERE Код_учня = 2;

Для того, щоб не слідкувати при добавленні та зміні даних за значеннями первинного ключа, в базі даних Access, в якості первинного ключа можна створити поле "Лічильник", тип поля вибрати "Автонумерація". Тоді значення поля "Лічильник" створюється автоматично під час кожного збереження запису.

Обмеження на допустимі значення для стовпця таблиці призначені для підтримки доменної цілісності. Ризик порушити доменну цілісність виникає при додаванні і оновленні записів. Обмеження доменної цілісності можна задати при створенні таблиці. Множину допустимих значень стовпця обмежує тип даних. Крім того, можна накласти на стовпець додаткові обмеження. Наприклад, якщо в стовпці "Дата_народження" таблиці "Учень" вказано значення #01.01.1970#, то ми не засумніваємося, що це помилкове значення. У цій же таблиці символьний стовпець "Стать" може приймати тільки два визначені значення, а не довільну комбінацію символів. Тобто, якщо ми спробуємо добавити дані, або змінити дані на такі, які не задовольняють обмеженням, що накладаються на дані в стовпці, то такі запити не зможуть виконатися.

У базі даних, яка добре спроектована, міститься декілька таблиць, які пов'язані між собою. Досить складні бази даних проектуються шляхом розробки множини окремих таблиць з подальшою установкою зв'язків між ними. Зв'язки між таблицями встановлюються за допомогою зовнішніх ключів (FOREIGN KEY). Так стовпець в одній таблиці може посилатися на стовпець іншої таблиці цієї ж бази даних. Подібні посилання являють собою обмеження посилкової цілісності бази даних і відіграють важливу роль при підтримці її загальної цілісності. Зв'язки між таблицями зазвичай несиметричні: одна таблиця залежить від іншої. Припустимо, у базі даних "Школа" є дві таблиці:

- Учні (Код_учня, Прізвище, Ім'я, Дата народження, Стать) – містить список учнів (Рис. 1);
- Оцінки (Код_оцінки, Предмет, Код_учня, Оцінка) – відомості про оцінки, які отримав учень з певного предмету (Рис. 2).

Оцінки

Код_оцінки	Код_учня	Код_предм	Оцінка
1	1	2	10
2	1	1	12
3	2	3	9
4	2	2	7
5	3	1	6

Рис. 2. Вміст таблиці "Оцінки" бази даних "Школа"

У таблиці "Учні" стовпець "Код_учня" є первинним ключем, тобто його значення визначені й унікальні (такі, що не повторюються). У таблиці "Оцінки" стовпець "Код_учня" не зобов'язаний мати унікальні значення, оскільки один і той же учень може мати декілька оцінок. Дані таблиці знаходяться в батьківсько-дочірньому відношенні: таблиця "Учні" - батьківська, "Оцінки" – дочірня. Даний зв'язок між таблицями організується шляхом оголошення стовпця "Код_учня" таблиці "Оцінки" зовнішнім ключем (FOREIGN KEY), який посилається на первинний ключ (PRIMARY KEY) "Код_учня" таблиці "Учні".

Використання зовнішніх ключів забезпечує збереження посилкової цілісності бази даних при зміні і видаленні записів. Разом з тим, наявність посилань породжує так звану проблему аномалій модифікації даних.

Якби таблиці "Учень" і "Оцінки" не були пов'язані, то при видалені записів з таблиці "Учень" у таблиці "Оцінки" могли залишитися посилання на учня, про якого вже немає відомостей. Цей факт зазвичай розцінюється як аномалія видалення. У випадку визначення в таблиці "Оцінки" зовнішнього ключа "Код_учня" з таблиці "Учень" не вдасться видалити учня, якщо він отримав хоча б одну оцінку.

Наступний запит не виконається:

DELETE * FROM Учні WHERE Код_учня = 3;

Якщо потрібно видалити з бази даних все, що стосується певного учня, то спочатку видаляються записи в таблиці "Оцінка",

DELETE * FROM Оцінка WHERE Код_учня = 3;

а потім – з таблиці " Учні ".

DELETE * FROM Учні WHERE Код_учня = 3;

Аналогічна ситуація може статися і при оновленні або добавленні даних. Наприклад, у таблиці "Учень" спробувати змінити ідентифікатор учня, який має оцінки в таблиці "Оцінка" і, таким чином, зовнішньому ключу тепер немає на що посилатися. Це аномалія зміни. В даному випадку необхідно спочатку додати новий запис в таблицю "Учень", вказавши в ній необхідне значення "Код_учня", потім змінити в таблиці "Оцінка" всі старі значення "Код_учня" на ті, які були введені новому записі таблиці "Учень", а потім видалити з таблиці "Учень" запис зі старим "Код_учня".

Щоб в таблицях, пов'язаних зовнішнім ключем, не робити модифікацію даних в декілька етапів, в базі даних Access при створенні зв'язків, у вікні "Знаряддя бази данних/Зв'язки" потрібно поставити галочку "Каскадне оновлення даних", "Каскадне видалення пов'язаних полів" (Рис. 3.) Тоді при видаленні або оновлені даних з таблиці "Учень" всі пов'язані дані з таблиці "Оцінки" видаляться, або оновляться.

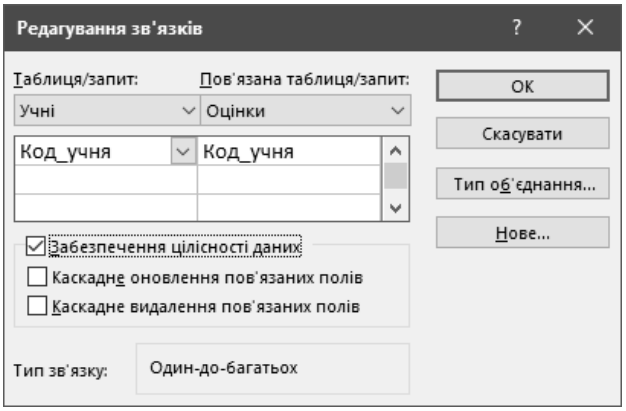

Рис. 3. Вікно редагування зв'язків в Access.

Щоб уникнути труднощів, при виконанні операцій добавлення даних в таблицю, їх зміни і видалення засобами мови запитів SQL здобувачі загальної середньої освіти повинні розуміти поняття первинного ключа, поняття обмеження цілісності відношень в реляційних базах даних та принципи організації зв'язків між таблицями. Створення зв'язків між таблицями часто сприймається здобувачами поверхнево, але воно є досить принциповим для розуміння роботи реляційної бази даних. Не розуміючи механізму зв'язків, забезпечення цілісності даних, здобувачі на практиці не можуть зв'язати одне поле з іншим, не можуть правильно побудувати запит, не розуміють, чому не виконується запит, який правильно структурно написаний. Більш детально про проблеми вивчення зв'язків у базах даних Microsoft Access описано в роботі [4].

Також труднощі виникають при розумінні написання запитів на вибірку даних з декількох таблиць.

Щоб правильно писати запити на вибірку даних з декількох таблиць, здобувачі загальної середньої освіти повинні розуміти принципи побудови та порядок виконання запитів SQL. SQL-вираз для вибірки даних, має вигляд [5]:

SELECT списокСтовбців
FROM списокТаблиць
WHERE УмовиПошуку
GROUP BY списокстовбцівГрупування
HAVING УмовиПошуку
ORDER BY УмовиСортування;

Оператори SELECT (вибрати) і FROM (з) в SQL-виразі, що визначають вибірку даних, є обов'язковими. Для уточнення запиту на вибірку даних існують додаткові оператори:

- WHERE (де) – вказує записи, які повинні увійти в результатну таблицю;
- GROUP BY (групувати по) – групує записи по значеннях певних стовпців;

- HAVING (що мають, за умови) – вказує групи записів, які мають увійти до результатної таблиці;
- ORDER BY (сортувати по) – впорядковує записи.

Ці оператори не є обов'язковими. Їх можна зовсім не використовувати, або використовувати лише деякі з них, або всі одразу. Якщо застосовуються декілька операторів, то в SQL-виразі вони виконуються у певному порядку.

Порядок перерахування операторів в SQL-виразі не співпадає з порядком їх виконання. Однак знання порядку виконання операторів допоможе здобувачам загальної середньої освіти уникнути багатьох непорозумінь при побудові запитів. Отже, перераховані оператори SQL-виразу виконуються в наступному порядку, передаючи один одному результат у вигляді таблиці [5]:

1. FROM – вибирає таблицю з бази даних; якщо зазначено декілька таблиць, то виконується їх декартовий добуток і результуюча таблиця передається для обробки наступному оператору.
2. WHERE – з таблиці вибираються записи, що відповідають умові пошуку.
3. GROUP BY – створюються групи записів, відібраних за допомогою оператора WHERE (якщо він присутній в SQL-виразі); кожна група відповідає якому-небудь значенню стовпця групування.
4. HAVING – обробляє кожну із створених груп записів, залишаючи тільки ті з них, які задовольняють умові пошуку;
5. SELECT – вибирає з таблиці, отриманої в результаті використання перерахованих операторів, тільки вказані стовпці.
6. ORDER BY – сортує записи таблиці.

Якщо нам потрібно вибрати всі відомості про учнів (всі поля таблиці "Учень") і всі відомості про оцінки (всі поля таблиці "Оцінки"), то ми можемо написати запит на вибірку:

SELECT Учні.*, Оцінки.* FROM Учні, Оцінки;

В результаті ми отримаємо декартовий добуток таблиць Учень і Оцінки (табл.3). Декартовим добутком двох реляційних відношень називається відношення яке містить усі можливі з'єднання рядків першого відношення з рядками другого відношення. Тобто, в нашому випадку ми отримаємо таблицю вміст якої зображену на рис.4.

Учні.Код_учня	Прізвище	Імя	Дата_народження	Стать	Код_оцінки	Оцінки.Код_учня	Код_предмета	Оцінка
1	Шпак	Максим	11.12.2009	ч	1	1	2	10
2	Григорчук	Петро	05.05.2009	ч	1	1	2	10
3	Хоменко	Ірина	07.10.2007	ж	1	1	2	10
1	Шпак	Максим	11.12.2009	ч	2	1	1	12
2	Григорчук	Петро	05.05.2009	ч	2	1	1	12
3	Хоменко	Ірина	07.10.2007	ж	2	1	1	12
1	Шпак	Максим	11.12.2009	ч	3	2	3	9
2	Григорчук	Петро	05.05.2009	ч	3	2	3	9
3	Хоменко	Ірина	07.10.2007	ж	3	2	3	9
1	Шпак	Максим	11.12.2009	ч	4	2	2	7
2	Григорчук	Петро	05.05.2009	ч	4	2	2	7
3	Хоменко	Ірина	07.10.2007	ж	4	2	2	7
1	Шпак	Максим	11.12.2009	ч	5	3	1	6
2	Григорчук	Петро	05.05.2009	ч	5	3	1	6
3	Хоменко	Ірина	07.10.2007	ж	5	3	1	6

Рис. 4. Результат виконання декартового добутку таблиць
"Учні" і "Оцінки" бази даних "Школа"

Аналізуючи результат виконання запиту, ми бачимо, що в деяких рядках містяться несумісні дані: в одному рядку дані про одного учня і дані про оцінки іншого учня. Для того, щоб отримати коректні дані, в результуючий набір нам потрібно взяти тільки відповідні дані, тобто ті дані де поле "Код_учня" з таблиці " Учень" рівне значенню "Код_учня" таблиці "Оцінка". Для цього ми відфільтруємо отримані дані за допомогою ключового слова WHERE.

SELECT Учні.*, Оцінки.* FROM Учні, Оцінки WHERE Учні.Код_учня=Оцінки.Код_учня;

Результат виконання даного запиту зображено на рис. 5.

Учні.Код_учня	Прізвище	Імя	Дата_народження	Стать	Код_оцінки	Оцінки.Код_учня	Код_предмета	Оцінка
1	Шпак	Максим	11.12.2009	ч	1	1	2	10
1	Шпак	Максим	11.12.2009	ч	2	1	1	12
2	Григорчук	Петро	05.05.2009	ч	3	2	3	9
2	Григорчук	Петро	05.05.2009	ч	4	2	2	7
3	Хоменко	Ірина	07.10.2007	ж	5	3	1	6

Рис. 5. Результат виконання запиту на вибірку інформації про учнів і їх оцінки.

Цей же результат можна отримати, використовуючи операції внутрішнього з'єднання таблиць (INNER JOIN) і ключового слова ON, за яким слідує умова відбору записів.

SELECT Учні.*, Оцінки.* FROM Учні INNER JOIN Оцінки ON Учні.Код_учня=Оцінки.Код_учня;

В даному випадку в результуючий набір з'єднуються тільки ті записи, в яких пов'язані поля обох таблиць співпадають. Внутрішні з'єднання відображають записи в результуючому наборі, як один запис. Якщо не знайдено відповідного значення у полі зв'язку пов'язаної таблиці, дані не відображаються взагалі (рис. 5). Незважаючи на те, що з'єднання з оператором WHERE є коротшим, краще використовувати оператор внутрішнього з'єднання INNER JOIN. Оператори SQL-виразу виконуються в певному порядку, передаючи один одному результат у вигляді таблиці. При використанні для з'єднання оператора WHERE, спочатку виконується оператор "FROM Учні, Оцінки", тобто виконується декартовий добуток двох відношень. Результат декартового добутку передається оператору WHERE, який відфільтровує не відповідні записи. Якщо припустити, що даних в таблицях буде багато і таблиць може теж бути більше ніж дві, то ця проміжна таблиця, яка передаватиметься оператору WHERE буде дуже великою на відміну від таблиці утвореної оператором INNER JOIN, який зразу в результуючу таблицю відбирає тільки відповідні дані. Тобто використання INNER JOIN для з'єднання двох і більше таблиць більш ефективне по відношенню з використанням оператора WHERE.

Висновки та перспективи подальшого дослідження. В роботі розглянуто складності у розумінні окремих понять та теоретичних положень з якими стикаються здобувачі загальної середньої освіти при вивченні теми "мова запитів SQL" розділу "Бази даних" в профільному курсі інформатики закладів загальної середньої освіти. Наведено відомості про поняття цілісності даних в реляційних базах даних і приклади написання запитів до бази даних, які ілюструють аномалії додавання, видалення та оновлення даних, що призводять до порушення цілісності даних. В роботі, також розглянуто порядок виконання операторів в запитах на вибірку даних. Знання порядку виконання операторів допоможе здобувачам загальної середньої освіти уникнути багатьох непорозумінь при побудові запитів. Вивчення поняття цілісності даних та розуміння принципів виконання запитів розвиває логічне мислення, допомагає засвоїти та закріпити навички конструювання запитів, привчає до контролювання можливих помилок при розв'язуванні практичних завдань.

Список бібліографічного опису

1. Інформатика. Профільний рівень // Навчальні програми для 10-11 класів [Електронний ресурс] — Режим доступу : https://mon.gov.ua/ua/osvita/zagalna-serednya-osvita/navchalni-programi/navchalni-programi-dlya-10-11-klasiv
2. Інформатика. Рівень стандарту // Навчальні програми для 10-11 класів [Електронний ресурс] — Режим доступу : https://mon.gov.ua/ua/osvita/zagalna-serednya-osvita/navchalni-programi/navchalni-programi-dlya-10-11-klasiv
3. Завадський І.О. Основи баз даних : навч. посіб. К. : Видавець І.О. Завадський, 2011. 192 с.
4. Шамшина Н.В. Методичні особливості вивчення зв'язків та типів об'єднання у базах даних Microsoft Access. Фізико-математична освіта. 2018. Випуск 1(15). С. 339–343.
5. Дунаев В. В. Базы данных. Язык SQL. СПб. : БХВ-Петербург, 2006. 288 с.

References

1. Informatyka. Profilnyy riven. Navchalni programi dlya 10-11 klasiv [Elektronnyy resurs]. URL: https://mon.gov.ua/ua/osvita/zagalna-serednya-osvita/navchalni-programi/navchalni-programi-dlya-10-11-klasiv (in Ukrainian)
2. Informatyka. Riven standartu. Navchalni programi dlya 10-11 klasiv [Elektronnyy resurs]. URL: https://mon.gov.ua/ua/osvita/zagalna-serednya-osvita/navchalni-programi/navchalni-programi-dlya-10-11-klasiv (in Ukrainian)
3. Zavadsky I.O. Fundamentals of the databases: textbook. K. : Publ. I.O. Zavadsky, 2011. 192 p. (in Ukrainian)
4. Shamshina N. Methodical Features Of Studying Relationships And Types Of Joins In Databases Microsoft Access. Physical and Mathematical Education. 2018. Issue 1(15). P. 339–343. (in Ukrainian)
5. Dunaev V.V. Database. SQL language. St. Petersburg: BHV- Petersburg, 2006. 288 p. (in Russian)

Рецензент: Мартинюк Олександр Семенович, доктор педагогічних наук, професор кафедри експериментальної фізики та інформаційно-вимірювальних технологій Східноєвропейського національного університету імені Лесі Українки,.