УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»
Коледж економіки, права та інформаційних технологій
Циклова комісія з інформаційних технологій

ДОБРИШИН Ю.Є., САСІМ М.О.

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ЩОДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ З ДИСЦИПЛІНИ «БАЗИ ДАНИХ»

(для студентів спеціальностей 121 «Інженерія програмного забезпечення»
та 122 «Комп'ютерні науки»)

Київ – 2019 р.

УДК 004.658.2

Розглянуто на засіданні циклової комісії з інформаційних технологій протокол № 1 від «28» серпня 2019 р. Рекомендовано до видання методичною радою Коледжу економіки, права та інформаційних технологій «Університет економіки та права «КРОК» протокол № 1 від «30» серпня 2019 р.

Автори: 1. Ю.Є. Добришин, кандидат технічних наук, доцент кафедри комп'ютерних наук Навчально-наукового інституту інформаційних та комунікаційних технологій «Університет економіки та права «КРОК».

2. М.О. Сасім, старший викладач Коледжу економіки, права та інформаційних технологій «Університет економіки та права «КРОК».

[Текст]: Методичні рекомендації щодо виконання лабораторних з дисципліни **БАЗИ ДАНИХ** [Ю.Є. Добришин, Сасім М.О.]; Університет економіки та права «КРОК» – Київ - 2019. – 44 с.

Методичні рекомендації для виконання лабораторних робіть містять практичні питання з тем навчальної дисципліни **«БАЗИ ДАНИХ»**, визначають технологію та особливості виконання технологічних операцій зі створення, додавання компонентів бази даних з використанням програмного забезпечення MS Access та MS SQL Server.

Видання призначене для студентів спеціальностей 121 «Інженерія програмного забезпечення» та 122 «Комп'ютерні науки»

РОЗГЛЯНУТО І СХВАЛЕНО

Педагогічною радою Коледжу економіки, права та інформаційних технологій
Протокол № 1 від «30» серпня 2019 р.

УДК 004.658.2

ЗМІСТ

ВСТУП

Метою виконання лабораторних занять з дисципліни «БАЗИ ДАНИХ» ставиться формування у слухачів практичних навиків з виконання операцій зі створення, редагування структурних компонентів бази даних, формування та аналізу запитів за її масивами, здійснення робіт щодо виконання тригерів та процедур, забезпечення обмеження цілісності інформаційних масивів.

За результатами виконання лабораторних робіт студенти повинні вміти:

1. Здійснювати операції щодо створення таблиць та побудови схеми бази з визначенням зв’язків між даними за допомогою відповідних ключових полів.
2. Виконувати операції з відбору даних з таблиць бази, здійснювати запити з використанням арифметичних та логічних операторів, пошукових запитів тощо.
3. Створювати бази даних та таблиці, логічну модель бази даних (ER діаграму) за допомогою програмного забезпечення MS SQL Server.
4. Виконувати операції з існуючими операторами мови Transact – SQL, які призначені для маніпулювання даними в таблицях бази СКБД MS SQL Server.
5. Виконувати операції зі створення процедур та тригерів з використанням операторів MS SQL Server.

КРИТЕРІЇ ОЦІНКИ ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Основа мета перевірки виконання лабораторних занять – виявлення здатності студента застосовувати одержані теоретичні знання на практиці.

Оцінка за виконання заняття ставиться як середньоарифметична суми оцінок безпосередньо за виконану роботу та захист.

Оцінка “відмінно” ставиться, якщо результати виконання роботи збігаються з результатами контрольного прикладу, завдання до лабораторної роботи виконані в повному обсязі, студент демонструє знання про матеріали роботи на рівні 90–100 %.

Оцінка “добре” – якщо результати виконання роботи частково збігаються з результатами контрольного прикладу, завдання до роботи виконані в повному обсязі, але студент демонструє знання матеріалів практичної роботи на рівні 75–90 %.

Оцінка “задовільно” – якщо результати виконання роботи частково збігаються з результатами контрольного прикладу, завдання до роботи виконані не в повному обсязі, студент демонструє знання наведеного матеріалу роботи на рівні 50–75 %.

Оцінка “незадовільно” – якщо студент не виконав завдання, що зазначені у роботі, не відповідає на теоретичні питання, які відносяться до теми роботи.

ЛАБОРАТОРНА РОБОТА №1

Створення таблиць та зв’язків між ними

Метою зазначеної роботи є виконання операцій щодо створення таблиць та побудови схеми бази з визначенням зв’язків між даними за допомогою відповідних ключових полів.

Під час виконання роботи у якості програмного продукту використовується програмне забезпечення Microsoft Access.

Приклад виконання роботи

1. Запустіть Microsoft Access. Створіть базу даних **«Фірма»**. Співробітники даної організації працюють з клієнтами та виконують їх замовлення.

2. Необхідно створити 3 таблиці: **«Співробітники, Клієнти та Замовлення».**

Таблиця 1. - Співробітники

Ім’я поля	**Тип даних**
Код співробітника	Лічильник
Прізвище	Текстовий
Ім’я	Текстовий
По-батькові	Текстовий
Посада	Текстовий
Телефон	Текстовий
Адреса	Текстовий
Дата народження	Дата/Час
Заробітна платня	Грошовий
Фото	Об’єкт OLE
Ел_пошта	Гіперпосилання

Таблиця 2. - Співробітники

Ім’я поля	**Тип даних**
Код клієнта	Лічильник
Назва компанії	Текстовий
Адрес	Текстовий
Номер телефону	Текстовий

Факс	Числовий
Адреса електронної пошти	Гіперпосилання
Замітки	Поле MEMO

Таблиця 3. - Замовлення

Ім’я поля	**Тип даних**
Код замовлення	Лічильник
Код клієнта	Числовий
Код співробітника	Числовий
Дата розміщення	Дата/Час
Дата виконання	Дата/Час
Сума	Грошовий
Відмітка о виконанні	Логічний

3. Окремі таблиці, які містять інформацію по конкретній тематиці, необхідно зв’язати в єдину структуру бази даних. Для зв’язку таблиць необхідно задати ключові поля. Ключ складається з одного або декількох полів, значення яких однозначно визначають кожен запис в таблиці.

Найкращім полем, у якості ключового є «Лічильник», так як значення в даному полі є унікальними (тобто виключені повторення).

4. Відкрийте таблицю **Співробітники** в режимі Конструктора.

5. Натисніть правою кнопкою миші на полі **Код співробітника** і у контекстному меню виберіть команду **Ключове поле**. Якщо в таблиці необхідно встановити кілька ключових полів, то виділити їх можна, утримуючи клавішу **Ctrl**.

6. Для таблиці **Клієнти** встановіть ключове поле **Код клієнта**, а для таблиці **Замовлення – Код замовлення**

7. Таблиця **Замовлення** містить поля **Код співробітника** і **Код клієнта**. При їх заповненні можуть виникнути деякі труднощі, тому що не завжди вдається запам'ятати всі підприємства, з якими працює фірма, і всіх співробітників з номером коду. Для зручності можна створити списки, що розкриваються за допомогою Майстра підстановок.

8. Відкрийте таблицю **Замовлення** в режимі **Конструктора.**

9. Для поля **Код співробітника** виберіть тип даних **Майстер підстановок**.

10. У вікні виберіть команду «**Об'єкт «стовпець підстановки»** буде використовувати значення з таблиці або запиту» і клацніть кнопку **Далі.**

11. У списку таблиць виберіть таблицю **Співробітники** і клацніть кнопку **Далі.**

12. У списку Доступні поля виберіть поле **Код співробітника** і клацніть на кнопці зі стрілкою, щоб ввести поле в список Вибрані поля. Таким же чином додайте поля Прізвище та Ім'я та клацніть кнопку Далі.

13. Виберіть порядок сортування списку по полю **Прізвище**.

14. У наступному діалоговому вікні задайте необхідну ширину стовпців із списку.

15. Встановіть прапорець **Приховати ключовий стовпець** та натисніть кнопку Далі.

16. На останньому кроці **Майстра підстановок** замініть при необхідності напис для поля підстановок і клацніть на кнопці **Готово**.

17. Аналогічним чином створіть список, що розкривається для поля **Код клієнта.**

18. Після створення ключових полів можна приступити до створення зв’язків. Існує кілька типів відносин між таблицями:

a) при відношенні **«один-до-одного»** кожному запису ключового поля в першій таблиці відповідає тільки один запис у зв'язаному полі іншої таблиці, і навпаки.

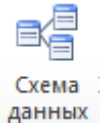
b) при відношенні **«один-до-багатьох»** кожному запису в першій таблиці відповідає декілька записів в другій, але запис у другій таблиці не може мати більше одного пов'язаного запису з першої таблиці;

c) при відношенні **«багато-до-багатьох»** одному запису в першій таблиці можуть відповідати кілька записів у другій таблиці, а одному запису в другій таблиці можуть відповідати кілька записів з першої.

19. Закрийте всі відкриті таблиці, так як створювати або змінювати

зв’язки між відкритими таблицями не можна.

20. Виконайте команду: вкладка **Робота з базами даних** → кнопка.



21. Якщо раніше ніяких зв'язків між таблицями бази не було, то при відкритті вікна Схема даних одночасно відкривається вікно Додавання таблиці, в якому виберіть таблиці **Співробітники, Клієнти та Замовлення**.

22. Якщо зв'язки між таблицями вже були задані, то для додавання в схему даних нової таблиці клацніть правою кнопкою миші на схемі даних і в контекстному меню виберіть пункт **Додати таблицю**.

23. Встановіть зв'язок між таблицями **Співробітники та Замовлення**, для цього виберіть поле **Код співробітника** в таблиці **Співробітники** і перенесіть його на відповідне поле в таблиці Замовлення.

24. Після перетягування відкривається діалогове вікно **Зміна зв'язків** (рис. 1), в якому включіть прапорець Забезпечення умови цілісності. Це дозволить запобігти випадкам видалення записів з однієї таблиці, при яких пов'язані з ними дані інших таблиць залишаться без зв'язку.

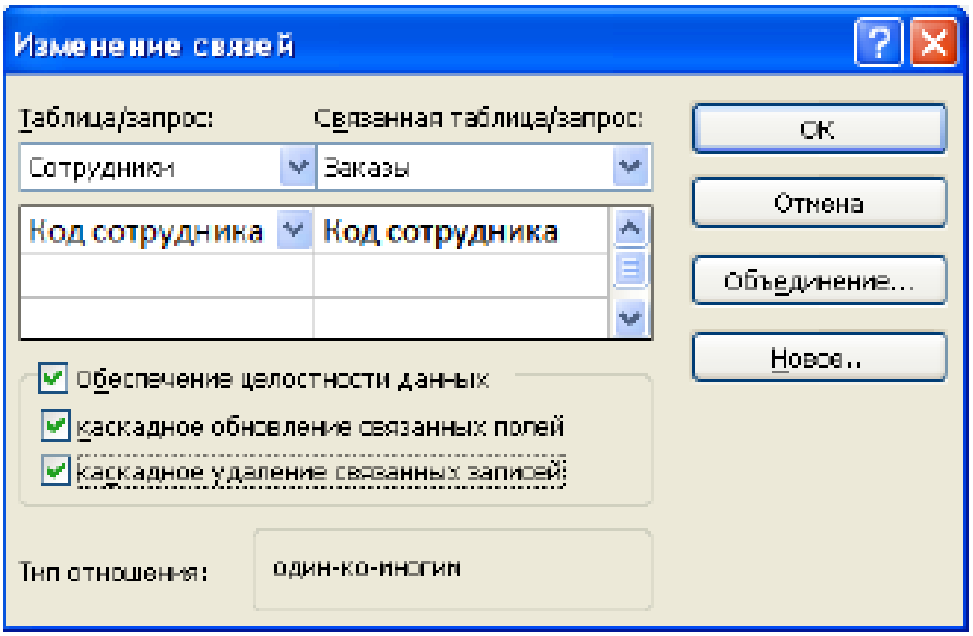


Рис. 1. Створення зв’язку між таблицями

25. Прапорці **Каскадне оновлення** зв'язаних полів і **Каскадне видалення** зв'язаних записів забезпечують одночасне відновлення чи видалення даних у всіх підпорядкованих таблицях при їх зміні в головній таблиці.

26. Параметри зв'язку можна змінити, натиснувши на кнопку Об'єднання.

27. Після встановлення всіх необхідних параметрів натисніть кнопку ОК.

28. Зв'язок між таблицями **Клієнти та Замовлення** встановіть самостійно.

29. В результаті повинна вийти схема даних, представлена на рис. 2.

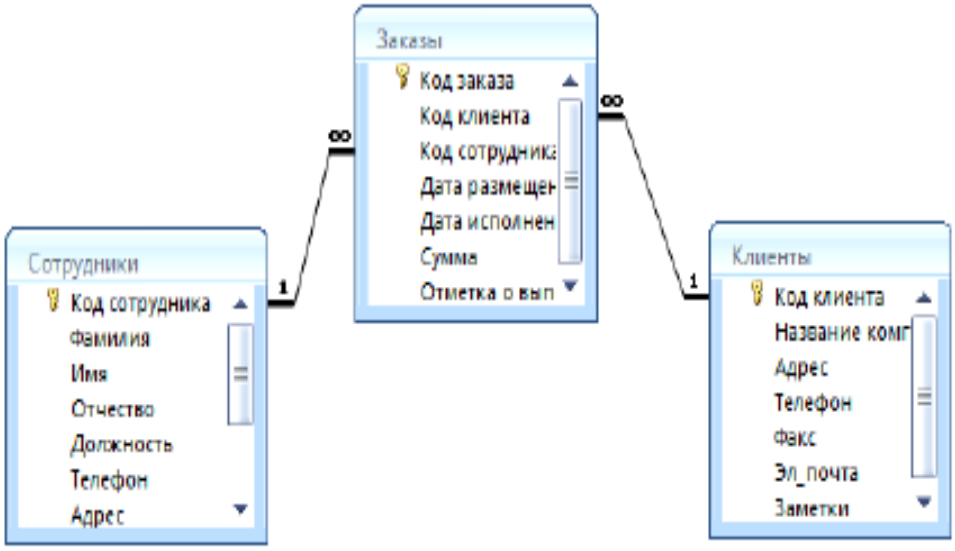


Рис. 2. Схема даних

У наведеному прикладі використовуються зв'язки «один-до-багатьох». На схемі даних вони відображаються у вигляді з'єднувальних ліній зі спеціальними значками поблизу таблиць. Зв'язок «один-до-багатьох» позначається «1» поблизу головної таблиці (що має первинний ключ) і «∞» поблизу підпорядкованої таблиці (що має зовнішній ключ). Зв'язок «один-до-одного» позначається двома «1» (обидва поля таблиць мають первинні ключі). Невизначений зв'язок не має ніяких знаків. Якщо встановлено об'єднання, то його напрямок відзначається стрілкою на кінці з'єднувальної лінії (жодне з об'єднаних полів не є ключовим і не має унікального індексу).

30. У таблицю Співробітники занести дані про сімох працівників.

31. У таблицю Клієнти занести дані про десять підприємств, з якими працює дана фірма.

32. У таблиці Замовлення оформіть кілька заявок, які надійшли на фірму.

Завдання на виконання лабораторної роботи

1. Для призначеного варіанта завдань за допомогою програмного забезпечення MS Access створити базу даних, три таблиці та схему даних.
2. Заповнити таблиці необхідними даними у відповідності з їх типом.
3. Відповісти письмово на контрольні питання.
4. Надати роботу на перевірку викладачу.

Варіанти завдань:

1. База даних «Страхова фірма». Орієнтовні таблиці: «Види страховок», «Клієнти \ об'єкти», «Страхова діяльність»

2 База даних «Агентство нерухомості». Орієнтовні таблиці: «Об'єкти нерухомості», «Продажі», «Покупки».

3 База даних «Деканат ВНЗ». Орієнтовні таблиці: «Список студентів», «Список предметів», «Сесія».

4 База даних відділу кадрів виробничого підприємства. Орієнтовні таблиці: «Співробітники», «Штатний розклад», «Відділи», «Цехи».

6 База даних фірми покупки і продажу автомобілів. Орієнтовні таблиці: «Продажі», «Покупки», «Автомобілі».

7 База даних «Готель» Орієнтовні таблиці: «Номери», «Рахунки», «Клієнти»

8 База даних «Розрахунок квартплати ТСЖ». Орієнтовні таблиці: «Список мешканців», «Оплати», «Тарифи».

9 База даних «Залізничні каси». Орієнтовні таблиці: «Продажі», «Посадочні місця», «Напрямки».

10 База даних «авіапасажирських перевезень» Орієнтовні таблиці: «Рейси», «Літаки», «Продажі».

11 База даних музею. Орієнтовні таблиці: «Експонати», «Автори», «Експозиції».

12 База даних «Спортивні комплекси району». Орієнтовні таблиці: «Нормативи», «Спортсмени», «Змагання»

13 База даних «Екзаменаційна сесія». Орієнтовні таблиці: «Предмети», Оцінки »,« Студенти ».

14. База даних «Турагентство». Орієнтовні таблиці: «Тури», «Продажі».

15 База даних Аптека ». Орієнтовні таблиці: «Товари», «Постачальники», «Продажі».

16 База даних «Збірка і реалізація комп'ютерів». Орієнтовні таблиці: «Продукція», «Клієнти», «Замовлення».

17 База даних Продуктові магазини району Орієнтовні таблиці: «Продажі», «Відділи», «Товари».

18. База даних лікарні (одного відділення). Орієнтовні таблиці: «Хворі», Діагнози »,« Лікарі »

19. База даних «Відеотека». Орієнтовні таблиці: «Артисти», «Фільми», «Продажі».

Контрольні питання:

1. Призначення, версії, характеристика програмного забезпечення MS Access.
2. Способи створення таблиць бази даних.
3. Створення таблиці бази у режимі конструктора.
4. Типи даних та їх коротка характеристика.
5. Поняття ключового полю. Типи ключів бази даних та їх призначення.
6. Поняття схеми даних. Особливості її побудови.
7. Поняття концептуальної моделі бази даних.
8. Поняття ER моделі бази даних.

ЛАБОРАТОРНА РОБОТА №2

Створення запитів за допомогою програмного забезпечення MS Access

Метою зазначеної роботи є виконання операцій з відбору даних з таблиць бази, виконання запитів з використанням арифметичних та логічних операторів, а також здійснення пошукових запитів та запитів з використанням «Построителя выражений».

Приклад виконання роботи

Запити є основним засобом перегляду, відбору, зміни та аналізу інформації, яка міститься в одній або кількох таблицях бази даних. Існують різні види запитів, але найбільш поширеними є **ЗАПИТИ НА ВИБІРКУ**.

1. Відкрийте базу даних **«Фірма»,** створену раніше.

2. Виконайте команду: **Створити → Майстер запитів → Простий запит.**

3. У діалоговому вікні (рис. 1) вкажіть таблицю **Співробітники** і виберіть поля **Прізвище, Ім'я, Телефон.** Натисніть кнопку **Далі**.

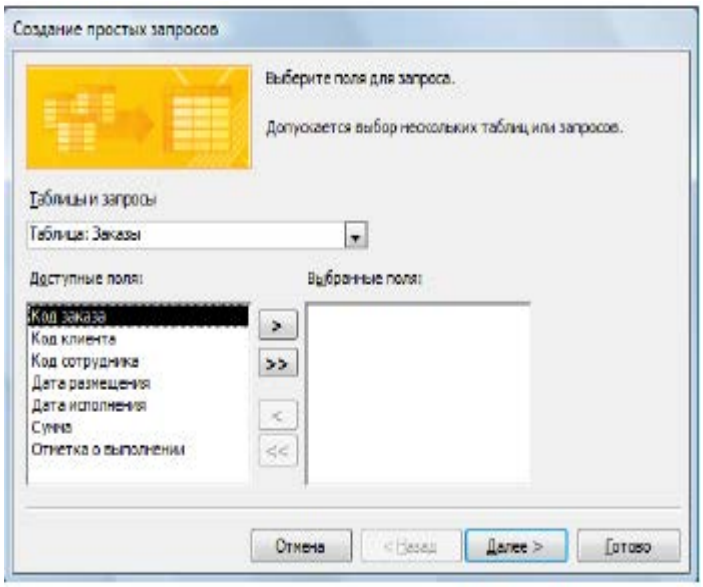


Рис. 1. Створення простого запиту

4. Введіть ім'я запиту – **Телефони** – і натисніть кнопку **Готово**. Перед вами з'явиться запит, в якому можна переглянути телефони співробітників.

5. Наступний запит спробуйте створити за допомогою Конструктора, для цього виконайте команду: **Створити → Конструктор запитів.**

6. У діалоговому вікні **Додавання таблиць** виберіть таблицю **Клієнти** і клацніть на кнопці **Додати**, а потім – на кнопці **Закрити**.

7. Щоб перенести потрібні поля в бланк запиту, необхідно по ним двічі клацнути лівою кнопкою миші (рис. 2).

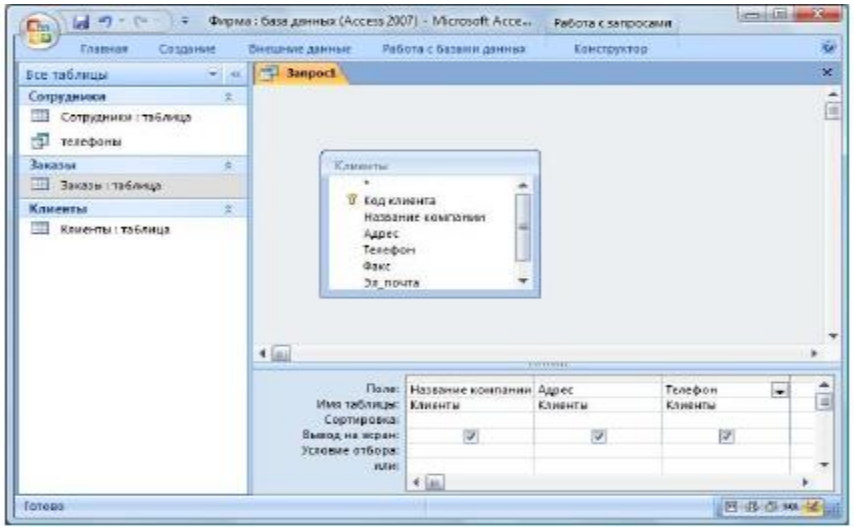


Рис. 2. Створення запиту в режимі Конструктора

8. Щоб відсортувати записи в полі Назва компанії в алфавітному порядку, необхідно в списку, що розкривається рядка Сортування вибрати пункт за зростанням.

9. Збережіть запит з ім'ям **«Адреси клієнтів».**

10. Самостійно створіть запит **«Дні народження»**, в якому можна буде переглянути дні народження співробітників.

11. Припустимо, ми хочемо дізнатися, у кого із співробітників день народження в поточному місяці, наприклад в квітні. Для цього відкрийте запит у режимі **Конструктора**.

12. У рядку Умова відбору для поля **«Дата народження»** введіть значення * .04. *. В даному записі * означають, що дата і рік народження можуть бути будь-якими, а місяць 4-м (тобто квітень). Після цього вікно запиту має виглядати так, як воно представлене на рис. 3.

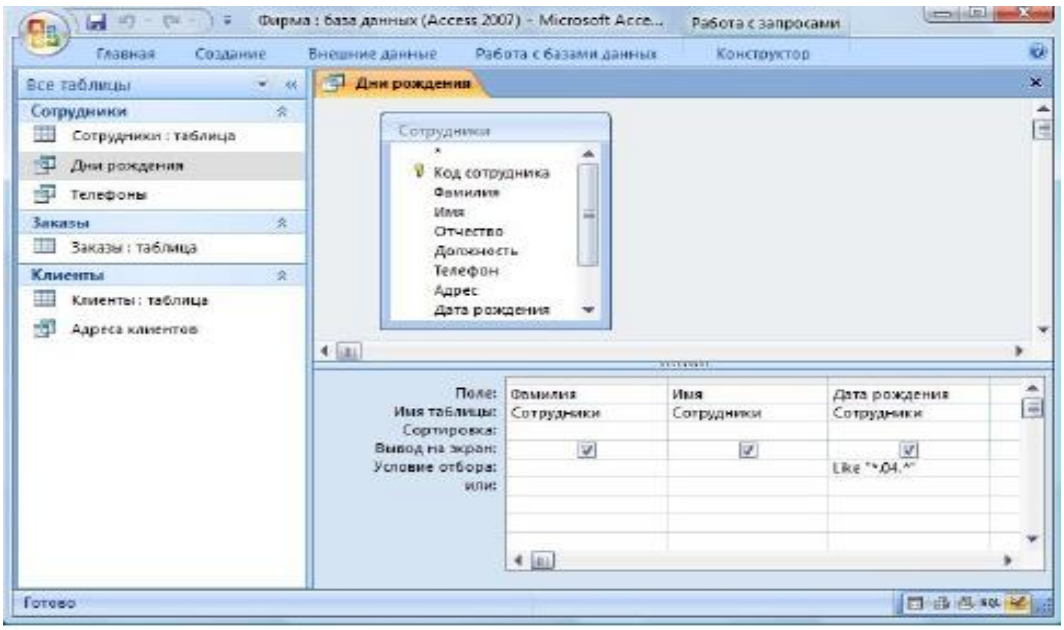


Рис. 3. Створення запиту

13. Закрийте **Конструктор** і перегляньте отриманий результат. Якщо в запиті Дні народження немає жодного запису, значить, у таблиці **Співробітники** немає жодної людини, народженої в квітні. Додайте в таблицю **Співробітники** декілька чоловік, що народилися в квітні, і подивіться, як зміниться запит. Запити автоматично оновлюються при кожному відкритті.

14. Якщо нам потрібно дізнатися, хто із співробітників народився в травні, то прийдеться створити новий запит або змінити умову в існуючому запиті **Дні народження**. Дана процедура є незручною і займає багато часу. Якщо доводиться часто виконувати запит, але кожен раз з новими значеннями умов використовують **ЗАПИТ З ПАРАМЕТРОМ**. При запуску такого запиту на екран виводиться діалогове вікно для введення значення в якості умови відбору. Щоб створити запит з параметром, користувачеві необхідно ввести текст повідомлення в рядку Умова відбору бланка запиту (рис. 4).

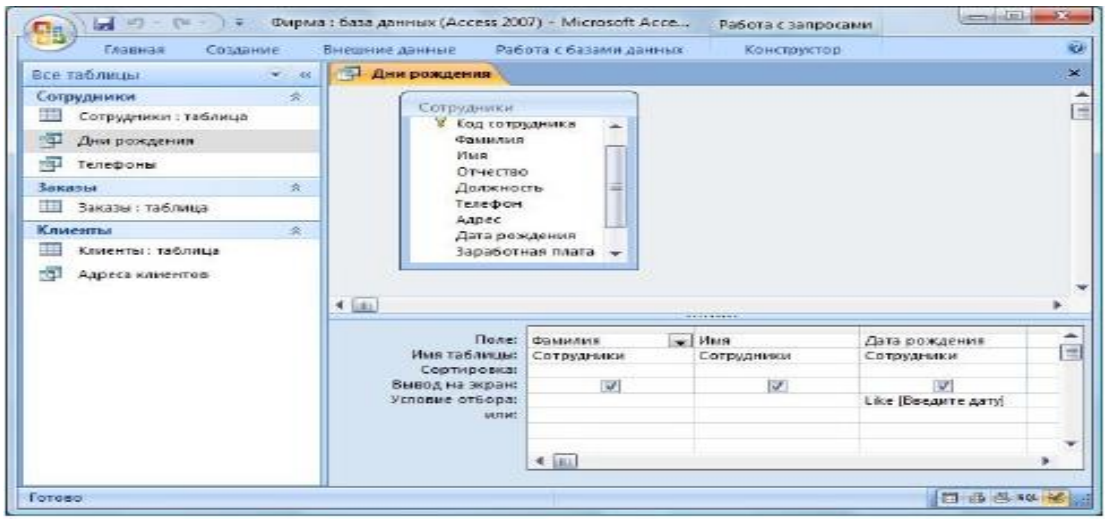


Рис. 4. Створення запиту з параметром

15. Запис **Like [Введіть дату]** означає, що при відкритті запиту з’явиться діалогове вікно (рис. 5) з текстом **«Введіть дату»** та полем для введення умови відбору. Якщо ввести умову * .04. *, то в запиті з'явиться список співробітників, які народилися в квітні. Запустіть запит ще раз і введіть значення * .05. *, подивіться, як змінився запит.

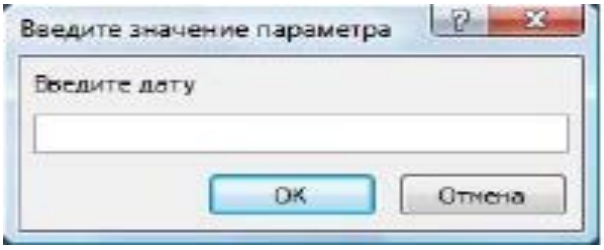


Рис. 5. Вікно для введення умови відбору

16. Змініть запит **«Телефони»** так, щоб при його запуску виводилося діалогове вікно з повідомленням «Введіть прізвище». Оскільки у запиті потрібно вивести конкретне прізвище, в умові відбору слово **Like** писати не треба.

17. Змініть запит **«Телефони»** так, щоб при його запуску запитувалось не тільки прізвище, але й ім'я співробітника.

18. Самостійно створіть запит **«Виконані замовлення»,** що містить такі відомості: прізвище та ім'я співробітника, назва компанії, в якій він працює, відмітка про виконання і сума замовлення. Дані запиту візьміть з декількох таблиць.

19. В умови відбору для логічного поля **Відмітка про виконання** введіть Так, щоб у запиті відображалися тільки виконані замовлення.

20. Зробіть так, щоб стовпчик Відмітка про виконання не виводився на екран.

21. Створіть запит Сума замовлення, в якому будуть відображатися замовлення на суму більше 50 000 грн.

22. Змініть запит, щоб сума замовлення була від 20 000 до 50 000 грн. Для даних запитів в Умові відбору можна використовувати оператори порівняння >, <, =,> =, <=, <> і логічні оператори And, Or, Not та ін.

23. Іноді в запитах потрібно зробити деякі обчислення, наприклад порахувати прибутковий податок 13% для кожної угоди. Для цього від-відкрийте запит Сума замовлення в режимі Конструктора.

24. У порожньому стовпці бланка запиту клацніть правою кнопкою миші на комірці Поле і в контекстному меню виберіть команду **Побудувати**. Перед вами з'явиться вікно «**Построитель выражений**» (рис. 6), який складається з трьох областей: поля виразу, кнопок операторів і елементів виразу. Зверху розташовується поле виразу, в якому воно і створюється. Елементи, які вводяться в це поле вибираються в двох інших областях вікна **Построителя.**

25. У лівому списку відкрийте папку Запити і виділіть запит Сума замовлення. У середньому списку виділіть поле Сума та натисніть кнопку Вставити. Ідентифікатор цього поля з'явиться в полі виразу Построителя.

26. Клацніть на кнопці * і введіть 0,13 (див. рис. 5). Таким чином, буде пораховано прибутковий податок 13%.

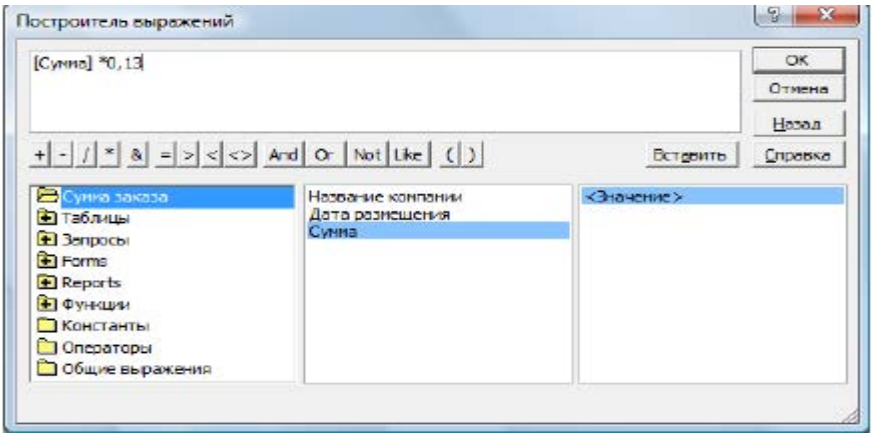


Рис. 5. Построитель выражений

27. Натисніть кнопку **ОК**, після чого в комірці **Властивості** Поля з'явиться значення «Вираз1: [Сума] * 0,13».

28. Замініть **Вираз1** на **Податок** і закрийте **Конструктор**.

29. Відкрийте запит і подивіться, що у вас вийшло.

30. Використовуючи **Построитель**, додайте в запит **Сума** замовлення поле Прибуток, в якому буде обчислюватися дохід від замовлення (тобто сума мінус податок).

31. Створіть запит **Менеджери**, за допомогою якого в таблиці **Співробітники** знайдіть всіх менеджерів фірми.

Завдання на виконання лабораторної роботи

1. Для призначеного варіанта завдань за допомогою програмного забезпечення MS Access створити запити за масивами бази даних.
2. Створити запит, що відбирає всі дані з таблиці.
3. Створити запит з параметрами.
4. Створити запити з логічними операторами **And, Or, Not.**
5. Створити запит з пошуковими параметрами.
6. Створити запит з використання Построителя з метою розрахунку параметрів у Вашої таблиці.
7. Надати роботу на перевірку викладачу.

Контрольні питання:

1. Способи створення запитів бази даних.
2. Створення запиту у режимі конструктора.
3. Призначення арифметичних операторів запиту.
4. Призначення логічних операторів запиту.
5. Оператори пошуку даних під застосування запиту.
6. Використання «Построителя выражений» для виконання запиту.

ЛАБОРАТОРНА РОБОТА №3

Створення бази даних та її таблиць, побудова ER діаграми

Метою зазначеної роботи є виконання операцій щодо створення бази даних та її таблиць, побудови логічної моделі бази даних (ER діаграми) за допомогою програмного забезпечення MS SQL Server.

Приклад виконання роботи

1. Створення бази даних

1.1. Створення бази даних у системі SQL-сервер здійснюється командою **CREATE DATABASE**. Слід зазначити, що процедура створення бази даних в SQL-сервері вимагає наявності прав адміністратора сервера.

Синтаксис щодо створення бази даних:

CREATE DATABASE –i'мя_бази_даних

[ON [PRIMARY]

[<визначення_файла> [,...n]]

[,<визначення_групи> [,...n]]]

[LOG ON {<визначення_файлу транзакцій>[,...n] }]

1.2. При створенні бази даних необхідно визначити набір файлів, з яких вона скрадатиметься. Файл визначається за допомогою наступної конструкції:

([NAME=логичне_i'мя_файлу,]

FILENAME='физичне_i'мя_файла' SIZE=розмер_файла]

[,MAXSIZE={max_розмер_файлу |UNLIMITED }]

[, FILEGROWTH=величина_приросту])[,...n]

Приклад 1. Створити базу даних, причому для даних визначити два файли на диску «D», для журналу транзакцій - один файл на диску «D».

```
CREATE DATABASE Archive
ON PRIMARY ( NAME=Arch1,
 FILENAME='d:\archdat1.mdf',
SIZE=100MB, MAXSIZE=200, FILEGROWTH=20),
(NAME=Arch2, FILENAME='d:\archdat2.mdf',
```

```
 SIZE=100MB, MAXSIZE=200, FILEGROWTH=20)
LOG ON
(NAME=Archlog1,
 FILENAME='d:\archlog1.ldf',
  SIZE=100MB, MAXSIZE=200, FILEGROWTH=20),
(NAME=Archlog2,
 FILENAME='d:\archlog2.ldf',
SIZE=100MB, MAXSIZE=200, FILEGROWTH=20);
```

2.Створення таблиці

2.1. Після створення загальної структури бази даних необхідно приступити до створення таблиць, які представляють собою відносини, що входять до складу проекту бази даних.

2.2. Таблиця - основний об'єкт для зберігання інформації в реляційної базі даних. Вона складається з містять строк і стовпців, займає в базі даних фізичний простір і може бути постійною або тимчасовою.

2.3. Базовий синтаксис оператора створення таблиці має наступний вигляд:

```
[USE имя_базы_данных ]
CREATE TABLE имя_таблицы
(имя_столбца тип_данных [NULL | NOT NULL ] [,...n])
```

Приклад: Створити таблицю для зберігання даних про товари, що надходять у продаж в деякій торговій фірмі. Необхідно врахувати такі відомості, як назва і тип товару, його ціна, сорт і місто, де товар виробляється.

```
USE ARCHIVE;
GO
CREATE TABLE TOVAR
(NAME VARCHAR(50) NOT NULL,
PRICE MONEY NULL,
TIP VARCHAR(50) NULL);
GO
```

Завдання на виконання лабораторної роботи

1. Здійснити концептуальне проектування бази даних та розробку ER моделі предметної області відповідно до наданого варіанту.
2. Перетворити концептуальну модель в реляційну базу даних.
3. За допомогою скриптів створити базу даних
4. Створити таблиці з назвами ставців та встановити обмеження.
5. Побудувати ER діаграму відповідно до створеної бази даних

Варіант 1. База даних комп’ютерна фірма

Схема бази даних складається з чотирьох таблиць:

Product (maker, model, type)

PC (code, model, speed, ram, hd, cd, price)

Laptop (code, model, speed, ram, hd, price, screen)

Printer (code, model, color, type, price)

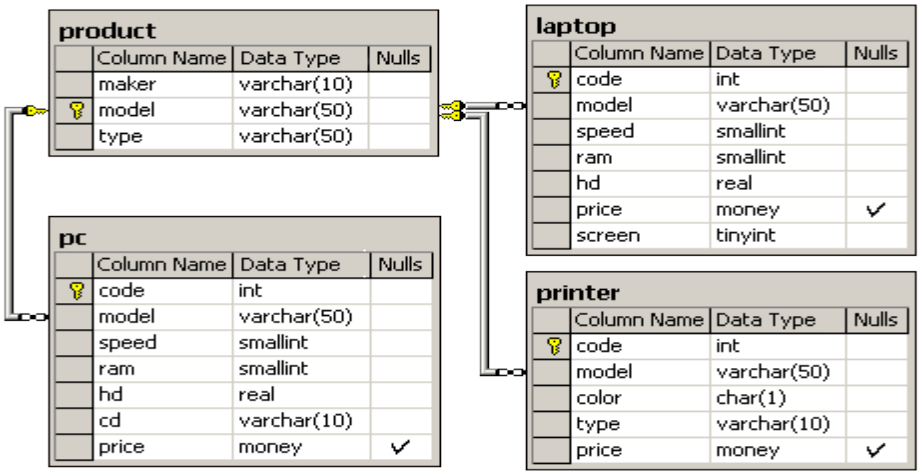
Таблиця Product представляє виробника (maker), номер моделі (model) і тип ('PC' - ПК, 'Laptop' - ПК-блокнот або 'Printer' - принтер).

Передбачається, що номери моделей в таблиці Product унікальні для усіх виробників і типів продуктів.

У таблиці PC для кожного ПК, однозначно визначається унікальний код, - code, вказані модель - model (зовнішній ключ до таблиці Product), швидкість - speed (процесора в мегагерцах), об'єм пам'яті - ram (у мегабайтах), розмір диска - hd (у гігабайтах), швидкість прочитуючого пристрою - cd (наприклад, '4x') і ціна - price.

Таблиця Laptop аналогічна таблиці PC за винятком того, що замість швидкості CD містить розмір екрану - screen (у дюймах).

У таблиці Printer для кожної моделі принтера вказується, чи є він кольоровим - color ('y', якщо кольоровий), тип принтера - type (лазерний - 'Laser', струминний - 'Jet' або матричний - 'Matrix') і ціна - price.



Варіант 2. Фірма вторинного продукту

Фірма має декілька пунктів прийому вторсировини. Кожен пункт отримує гроші для їх видачі здавальникам вторсировини.

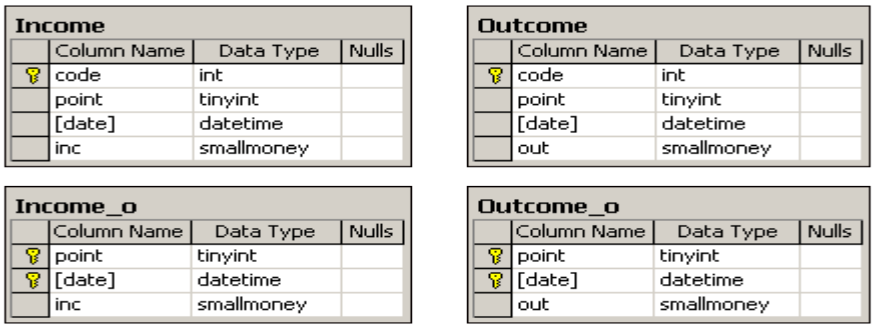
Відомості про отримання грошей на пунктах прийому записуються в таблицю: Income _ o (point, date, inc)

Первинним ключем є (point, date). При цьому в стовпець date записується тільки дата (без часу), тобто прийом грошей (inc) на кожному пункті робиться не частіше за один раз в день.

Відомості про видачу грошей здавальникам вторсировини записуються в таблицю: Outcome _ o (point, date, out)

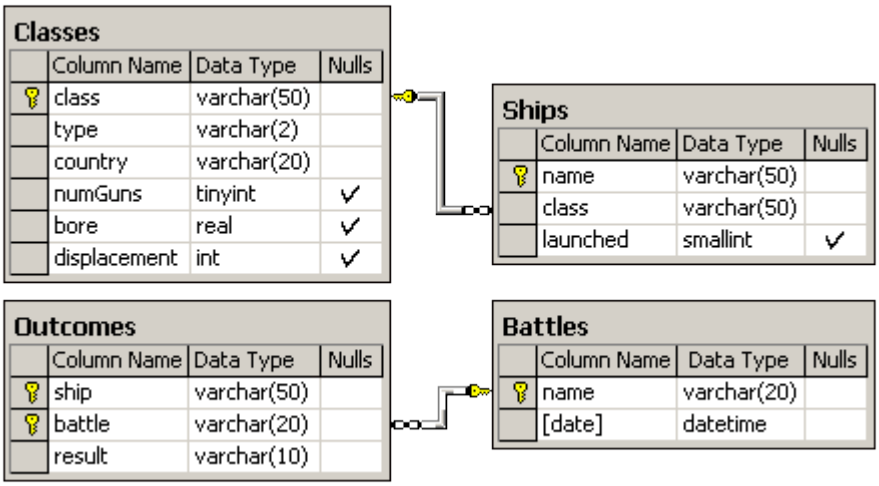
В цій таблиці також первинний ключ (point, date) гарантує звітність кожного пункту про видані гроші (out) не частіше за один раз в день.

У разі, коли прихід і витрата грошей може фіксуватися кілька разів в день, використовується інша схема з таблицями, що мають первинний ключ,code: Income (code, point, date, inc) Outcome (code, point, date, out) Тут також значення стовпця date не містять часу.



Варіант 3. Кораблі

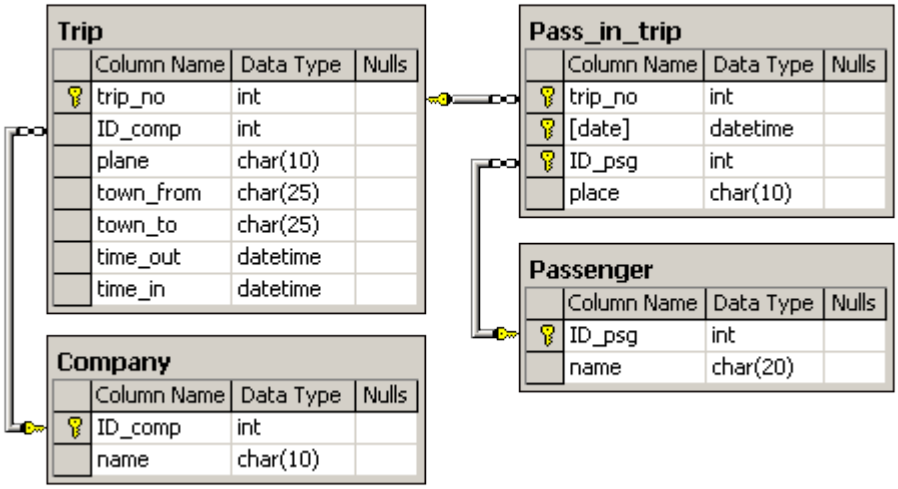
Розглядається база даних (БД) кораблів, що брали участь в другій світовій війні. Є наступні стосунки: Classes (class, type, country, numGuns, bore, displacement) Ships (name, class, launched) Battles (name, date) Outcomes (ship, battle, result) Кораблі в "класах" побудовані за одним і тому ж проектом, і класу привласнюється або ім'я першого корабля, побудованого за цим проектом, або назві класу дається ім'я проекту, яке не співпадає ні з одним з кораблів у БД. Корабель, що дав назву класу, називається головним. Відношення Classes містить ім'я класу, тип (bb для бойового (лінійного) корабля або bc для бойового крейсера), країну, в якій побудований корабель, число головних знарядь, калібр знарядь (діаметр ствола гармати в дюймах) і водотоннажність (вага в тоннах). Відносно Ships записані назва корабля, ім'я його класу і рік спуску на воду. У відношення Battles включені назва і дата битви, в якій брали участь кораблі, а відносно Outcomes - результат участі даного до



Варіант 4. Аерофлот

Схема БД складається з чотирьох відношень Company (ID _ comp, name) Trip (trip _ no, ID _ comp, plane, town _ from, town _ to, time _ out, time _ in) Passenger (ID _ psg, name) Pass _ in _ trip (trip _ no, date, ID _ psg, place) Таблиця Company містить ідентифікатор і назву компанії, що здійснює перевезення пасажирів. Таблиця Trip містить інформацію про рейси: номер рейсу, ідентифікатор компанії, тип літака, місто відправлення, місто прибуття, час

відправлення і час прибуття. Таблиця Passenger містить ідентифікатор і ім'я пасажира. Таблиця Pass _ in _ trip містить інформацію про польоти: номер рейсу, дата вильоту (день), ідентифікатор пасажира і місце, на якому він сидів під час польоту. При цьому слід мати на увазі, що - рейси виконуються щодня, а тривалість польоту будь-якого рейсу менше доби; town _ from <> town _ to; - час і дата враховується відносно одного часового пояса; - час відправлення і прибуття вказується з точністю до хвилини; - серед пасажирів можуть бути однофамільці (одинак



Контрольні питання:

1. Поняття концептуального проектування бази даних.
2. Логічна схема бази, принцип побудови ER діаграми бази, особливості її створення засобами MS SQL Server.
3. Класифікація типів даних MS SQL Server, числові дані та їх характеристика.
4. Символьні дані MS SQL Server, тип даних char, varchar основна різниця.
5. Обмеження, що призначаються для даних бази, типи ключових полів та їх характеристика.
6. Основні команди мови SQL (DDL) щодо створення та роботи з об’єктами бази даних.

ЛАБОРАТОРНА РОБОТА №4

Виконання операцій з даними таблиць

Метою зазначеної роботи є виконання операцій операторами мови Т – SQL, які призначені для маніпулювання даними в таблицях бази MS SQL Server.

Приклад виконання роботи

1. Оператор «Insert»

1.1. Оператор insert, призначений для додавання рядків в таблицю бази даних.

Синтаксис зазначеного оператора має дві форми запису:

1. **Insert Into <назва таблиці> ([<І'мя стовпця>, ...]) Values (<Значення>,...);**

2. **Insert Into <назва таблиці> Select <і'мя стовпця>,. From <назва таблиці> ;**

1.2. В разі використання першої форми тільки один рядок (або частина рядку) додається в таблицю. Друга форма запису передбачає додавання набору даних, який був отриманий за допомогою оператора «select». Приклади застосування оператора «Insert» для додавання даних в таблицю «employee»:

```
USE sample;
go
Insert Into employee Values (25348, 'Matthew', 'Smith', 'd3');
go
USE sample;
go
Insert Into employee Values (15201, 'Dave', 'Davis', Null);
go
```

1.3. Використання ключового параметра «Null» дозволяє додавати в рядок таблиці нулеві значення.

1.4. Друга форма оператора «Insert» додає в таблицю одну або декілька рядків, після їх отримання за результатами застосування запиту з використанням оператора «select»:

USE sample;

go

Create table dallas_dept

(dept_no CHAR (4) Not null,

dept_name CHAR (20) Not null);

go

Insert Into dallas_dept (dept_no, dept_name)

Select dept_no, dept_name From department

Where location = 'Dallas';

1.5. За результатами виконання запиту створюється нова таблиця «dallas_dept». Оператор «select» виконує відбір даних з таблиці «department» рядка «location», які відповідають значенню «Dallas», після чого оператор «Insert» додає отримані дані в рядки нової таблиці «dallas_dept».

2. Оператор «Update

2.1. Оператор **«Update»** виконує операції зміни даних в рядках таблиці. Синтаксис оператора має наступну форму:

Update tab_name Set column_1={expression}

Where {condition};

де, **«tab_name»** - ім'я таблиці бази даних, **«set»** - нове значення даних стовпчика, які будуть оновлені, оператор **«where»** - визначає умови відбору даних. Приклад виконання запита з оператором «Update»

USE sample;

go

Update works_on Set job = 'Manager'

Where emp_no = 18316;

go

2.2. У наведеному прикладі співробітник відділу, який має табельний номер 18316, призначається на посаду «Manager». В разі виконання змін всіх рядків таблиці запит не потребує застосування оператора «where» і може мати вигляд:

```
USE sample;
go
Update project Set budget = budget *0.51;
go
```

2.3. У приведеному прикладі змінюються дані всіх рядків таблиці **«project»**, тому що відсутній оператор **«where».** У разі відсутності конкретних умов відбору, бюджет всіх проектів буде збільшений на 0,51%.

3. Оператор «Delete»

3.1. Оператор «Delete» виконує видалення рядків з таблиці. Синтаксис оператора «Delete» має наступну форму запису:

```
Delete from table_name
[ Where predicate];
```

3.2. Якщо оператор **«where»** відсутній, будуть видалені **всі рядки** з таблиці. Приклад видалення всіх менеджерів з таблиці **«works_on»** бази даних «sample»

```
USE sample;
go
Delete from works_on
Where job = 'Manager';
go
```

3.4. Оператор **«delete»** може включати внутрішній запит з використанням операторів **«select»** та **«where»**.

4. **Оператор «Truncate Table»**

4.1. Оператор здійснює видалення всіх рядків таблиці, не використовуючи оператор **«Where»**. **Швидкість** виконання запиту за допомогою оператора **«Truncate Table»** значна краще ніж з оператором **«Delete»**, тому що під час застосування оператора **«Truncate Table»**, послідовно здійснюється видалення всіх сторінок таблиці. У відмінності від цього оператор **«Delete»** виконує видалення рядків таблиці. Синтаксис оператора **«Truncate Table»** має наступний вигляд:

Truncate table table_name;

Де, **table_name –** назва таблиці

На приклад:

Truncate table department;

5. **Речення OUTPUT**

5.1. Результат операцій **INSERT, UPDATE, DELETE** завжди містить тільки текст, в якому присутні відомості щодо кількості рядків, що були змінені. В разі явного відображення рядків, які були додані, змінені або видалені використовується речення **OUTPUT.** Приклад застосування речення **OUTPUT** з оператором **DELETE**

```
USE sample;
go
DECLARE @del TABLE (emp_no INT, emp_lname CHAR (20));
DELETE from employee UTPUT DELETED.emp_no, DELETED.emp_lname INTO @del WHERE emp_no > 15000;
Select * from @del;
go
```

5.2. В зазначеному прикладі використовується перемінна **@del_table** з двома стовпчиками: **Emp_no** та **emp_lname**. Синтаксис оператора **DELETE** розширений оцією **OUTPUT**

Output DELETED.emp_no, DELETED.emp_lname INTO @ del_table.

5.3. Система зберігає рядки в таблиці DELETED, яка потім копіюється в табличну переміну @del_table

6. Оператор MERGE

6.1. Оператор з'єднає послідовність команд **INSERT** та **UPDATE** в одному операторі. Оператор **MERGE** являє собою місце зберігання даних, де необхідно періодично оновлювати таблиці з урахуванням даних, від систем оперативної обробки транзакцій.

Приклад застосування оператора **MERGE:**

```
USE sample;
Create table Prodazhi
 (pr_no CHAR (4),
bonus int DEFAULT 100);
NSERT INTO Prodazhi (pr_no) VALUES (‘p1’);
```

6.2. Створюється таблиця **Prodazhi** s, яка містить рядок (‘p1’). Таблиця може бути використана для з'єднання

```
MERGE INTO Prodazhi P
  USING (SELECT project_no, budget
         from project) E
 ON (P.pr_no = E.project_no)
   WHEN MATCHED THEN
 UPDATE SET P.bonus = E.budget
   WHEN NOT MATCHED THEN
```

INSERT (pr_no, bonus)

Values (E.project_no, E.budget *0.5);

6.3. Оператор MERGE змінює данні в таблиці **Prodazhi** в залежності від існуючих значень в стовпчику **pr_no**. Якщо значення стовпця **project_no** таблиці **project** з'являється в стовпчику **pr_no** таблиці bonus, буде виконане **MATCHED,** в іншому буде виконане **NOT MATCHED** і оператор **INSERT** додає новий рядок в таблицю **bonus**.

Завдання на виконання лабораторної роботи

1. Виконати за допомогою оператора INSERT вставку 10 рядків в таблиці баз даних відповідно до варіанту завдань.
2. Створити за допомогою оператора UPDATE оновлення 6 даних в таблицях баз даних, відповідно до призначеного варіанту.
3. Виконати видалення 3 рядків у таблицях баз даних, відповідно до призначеного варіанту.
4. Створити службову таблицю, з назвою TOVAR. Заповнити її даними, далі виконайте видалення даних за допомогою оператора truncate table.
5. Для визначеного варіанту застосуйте речення OUTPUT з оператором DELETE
6. Виконати для таблиці бази оператор MERGE з метою оновлювання таблиці.

Контрольні питання:

1. Назвіть групу операторів мови SQL, які відносяться до операторів DML.
2. Особливості застосування оператора «select» з оператором «insert».
3. Застосування оператора update під час завдання умов оновлення, використовуючи умовний оператор «case».
4. В чому полягає різниця застосування оператора «delete» та «truncate table».
5. Призначення оператору MERGE та OUTPUT.

ЛАБОРАТОРНА РОБОТА №5

Створення запитів у MS SQL Server

Метою зазначеної роботи є створення запитів за інформаційними масивами бази MS SQL Server за допомогою операторів мови T-SQL.

Приклад виконання роботи

1. Оператор SELECT

1.1. Форма записи оператора «select», яка має наступний синтаксис:

select column_list from { table1 [tab_alias1]}, ……;

«Table1» - ім'я таблиці, інформація з якої запитується. Параметр **«tab_alias1»** є друге ім'я відповідної таблиці, що може бути використаний в якості скороченої її назви.

Параметр **«column_list»** містить один або декілька специфікацій:

- встановлює позначення «*», яке зазначає всі стовпчики таблиці, що визначена оператором **«from»**;

- встановлює запис «column_name as column_heading» з метою зміни ім'я стовпчика таблиці;

- встановлює явно ім'я стовпчиків необхідної таблиці;

- встановлює системні та агрегатні функції;

- встановлює математичні вирази.

Приклад:

1.2. Приведена форма запиту щодо відбору даних з таблиці за допомогою оператора «select»:

```
USE sample;
go
select dept_no, dept_name, location from department;
go
```

2. Умова WHERE

2.1. Умова **WHERE** фільтрує набір даних , який сформований блоком FROM, і вибирають з цього набору лише ті строки , які задовольняють умові. Умови можуть зсилатись на дані в таблицях, вирази , вбудовані скалярні функції

SQL і функції для користувачів. В умові WHERE можуть також використовуватись оператори порівняння і символи макропідстановки.

3. Використання умови BETWEEN

3.1. Умова **BETWEEN** перевіряє значення на його приналежність деякому діапазону. В даному випадку діапазон включить граничні значення. Наприклад : умова **between** 1 and 10 буде істиним для чисел 1 і 10

Найчастіше умову **BETWEEN** застосовують разом з датою.

Приклад:

```
USE Cha2;
go
SELECT EventCode, DateBegin FROM dbo.Event WHERE DateBegin BETWEEN ‘01/01/11’ AND ‘05/05/200 ;
go
```

4. Сортування даних за допомогою оператору ORDER BY

4.1. Найпростішим способом сортування результуючого набору даних є явно заданий порядок слідування стовбців, по яким відбувається упорядкування.

Приклад використання **«ORDER BY»** для отримання номерів відділів та імен користувачів, у яких номера менше ніж 20000

```
USE sample;
go
select emp_fname, emp_lname, dept_no from employee where emp_no < 20000 order by emp_lname;
go
```

5. Комбінація SELECT, DISTINCT

5.1. Першим предикатом, що використовується в поєднанні з ключовим словом **SELECT**, є **DISTINCT** . Він виключає дублювання даних запиту. Ці дублювання оцінюються на рівні стовбців результуючого набору даних а не початкових таблиць. Протилежну функцію виконує предикат ALL. Так як, він використовується за замовчуванням, у запитах його зазвичай ігнорують.

Приклад: Під час виконання запиту за масивами таблиці **«works_on»** бази даних «**sample**» стосовно користувачів, які працюють на проектом 1 або над проектом 2, був отриманий наступний результат:

USE sample;

go

select project_no, emp_no from works_on where project_no='p1'

or project_no='p2';

go

Результат запиту:

project_no emp_no

P1 10102

P2 25348

P2 18316

P2 29346

P1 9031

P1 28559

P2 28559

P1 29346

5.2. Приклад свідчить про наявність дублікатів значень стовпчика «emp_no». В такому випадку для усунення залишкової інформації, під час застосування запиту необхідно використовувати опцію **«DISTINCT»**

USE sample;

go

select DISTINCT emp_no from works_on where project_no='p1' or project_no='p2';

Результат запиту:

emp_no

10102, 25348, 18316, 9031, 28559, 29346

6. Оператор LIKE

6.1. Оператор «Like», призначений для перевірки відповідності значень певно му шаблону, сформованому в запиті за допомогою шаблонних символів. Форма запису оператора «Like» має наступний вигляд:

column Like ‘pattern’

6.2. Параметр **«pattern»** може бути константою типу рядок, дати, або використовувати наступні символи:

«%» (символ відсотка)- задає любу послідовність символів з нуля або більш;

«_» (символ підкреслення) – задає любій одиночний символ.

6.3. Приклад використання оператора «Like» для пошуку користувачів, у яких ім’я, прізвище має літеру «а» після другого символу

```
USE sample;
go
Select emp_frame, emp_lname, emp_no from employee where emp_frame Like ‘_a%’;
go
```

7. Основні підсумкові функції

7.1. Мова SQL містить велику множину підсумкових функцій, які можна використовувати в якості виразів в інструкції SELECT для отримання підсумкових даних. Функції , що найчастіше використовуються: **«min, max, sum, avg, count, count_big»**

Приклад синтаксиса запита «select» з функцією **«min»** та **max:**

```
USE sample;
go
select min (emp_no) as min_employee_no from employee;
go
USE sample;
go
select max (enter_date) from works_on where job =’Manager’;
```

Завдання на виконання лабораторної роботи

1. Створити базу даних ZAPASU
2. Створити таблиці бази даних.
3. Побудувати ER діаграму бази
4. За допомогою команди insert виконати заповнення таблиць даними.

Приклади таблиць.

Card (Картотека)

ID (ключове поле)	Qt (Кількісь)	Nom_ID Посилання на Nom(ID)	Store_ID Посилання на Store(ID)	Party_ID(NULL) Посилання на Party(ID)	Owner_ID(NULL) Властник- Посилання на Contractor (ID)	Customer_ID(NULL) Постачальник - Посилання на Contractor (ID)
1	10	1	1	NULL	1	NULL
2	20	2	1	1	NULL	1
3	25	3	1	NULL	2	NULL
4	45	6	2	2	NULL	2
5	7	2	2	NULL	1	3
6	17	1	2	3	1	NULL
7	45	7	2	3	NULL	4
8	34	3	3	3	3	5
9	6	2	3	1	3	NULL

Nom (Номенклатура) - список товарів

ID (ключове поле)	Nm
1	Стол
2	Стул
3	Шкаф
5	Тумбочка
6	Кресло
7	Ваза
8	Люстра

Party (Партия) - список партій

ID(Ключове поле)	Nm
1	Первая
2	Вторая
3	Третья

Contractor (Контрагент) - список Контрагентів

ID(Ключове поле)	Nm
1	ООО "Веллатрест"
2	ООО "Оникс"
3	ООО "ПРОГРЕСС"
4	ООО “Интеллект Сервис”
5	“УКРСПЕЦЭКСПОРТ”

Store (Склад)- список складів

ID(Ключове поле)	**Nm**
1	Готовая продукция
2	Товары на складе
3	Сырье и материалы

Виконати наступні запити:

1. Отримати з таблиці Card та Nom всі записи щодо колонок (ID, Qt,Nm).

2. Отримати з таблиць бази всі записи щодо колонок (ID, Qt, найменування власника, постачальника).

3. Отримати загальний список «Contractor» та складів «Store» за таблицями бази даних.

4. Отримати загальну кількість товару на складах окремо по номенклатурі та назву товару.

5. Отримати номенклатуру, яка не використовується у таблиці «Card».

Контрольні питання:

1. Надати перелік, призначення основних операторів, що застосовуються під час виконання простих запитів.

2. Особливості застосування опції DISTINCT та оператору LIKE.

3. Призначення та особливості застосування логічних операторів AND, OR, NOT,IN. BETWEEN.

4. Перелік основних агрегатних функцій MS SQL Server, особливості застосування функції COUNT.

5. Оператор GROUP BY та ORDER BY, порядок їх застосування.

ЛАБОРАТОРНА РОБОТА №6

Створення процедур і тригерів MS SQL Server.

Метою зазначеної роботи є виконання операцій щодо створення процедур та тригерів з використанням операторів MS SQL Server.

Приклад виконання роботи

1. Процедури

1.1. Збережені процедури представляють собою набір команд, що складається з одного або кількох операторів SQL, або функцій та зберігається в базі даних в відкомпілювався вигляді.

У SQL Server є кілька типів збережених процедур.

1.2. Системні процедури призначені для виконання різних адміністративних дій. Практично всі дії з адміністрування сервера виконуються з їх допомогою. Можна сказати, що системні збережені процедури є інтерфейсом, що забезпечує роботу з системними таблицями, яка, в кінцевому рахунку, зводиться до зміни, додавання, видалення і вибірці даних із системних таблиць як для користувача, так і системних баз даних. Системні процедури мають префікс «sp_», зберігаються в системній базі даних і можуть бути викликані в контексті будь-якої іншої бази даних.

1.3. Користувальницькі процедури реалізують ті чи інші дії. Збережені процедури - повноцінний об'єкт бази даних. Внаслідок цього кожна збережена процедура розташовується в конкретній базі даних, де і виконується.

1.4. Тимчасові процедури існують лише деякий час, після чого автоматично знищуються сервером. Вони діляться на локальні і глобальні.

1.5. Локальні тимчасові процедури можуть бути викликані тільки з того з'єднання, в якому створені. При створенні такої процедури їй необхідно дати ім'я, що починається з одного символу **#.** Як і всі тимчасові об'єкти, збережені процедури цього типу автоматично видаляються при відключенні користувача, перезапуск або зупинці сервера.

1.6. Глобальні тимчасові процедури доступні для будь-яких з'єднань сервера, на якому є така ж процедура. Для її визначення досить дати їй ім'я, що починається з символів # #. Видаляються ці процедури при перезапуску або зупинці сервера, а також при закритті з'єднання, в контексті якого вони були створені.

Приклад створення процедури, що змінює оклад співробітника.

```
Use sample;
go
Create procedure FFF (@oklad varchar(20) =5)
As
Begin
Update dbo.sotrydnik Set oklad =@oklad;
End
go
```

1.7. Запуск процедури здійснюється за допомогою команди **EXEC.** Приклад запуску процедури:

```
EXEC increase_budget;
```

Приклад: Створення процедури для зменшення ціни товару першого сорту на 10%

```
CREATE PROC my_proc2
AS
UPDATE Товар SET Ціна = Ціна * 0.9
  WHERE Сорт = 'перший'
```

Для звернення до процедури можна використовувати команду:

EXEC my_proc2

Приклад створення процедури з вхідним параметром.

1.8. Створити процедуру для отримання назв і вартості товарів, які придбав заданий клієнт.

```
CREATE PROC my_proc3
    @ K VARCHAR (20)
AS
SELECT Товар.Названіе,
    Товар.Цена * Сделка.Колічество
AS Вартість, Кліент.Фамілія
FROM Клієнт INNER JOIN
(Товар INNER JOIN Угода
ON Товар.КодТовара = Сделка.КодТовара)
ON Кліент.КодКліента = Сделка.КодКліента
WHERE Кліент.Фамілія = @ k
```

Для звернення до процедури можна використовувати команду:

```
EXEC my_proc3 'Іванов'
```

2. Тригери

2.1. Дії тригера пов'язані з використанням оператора DML (**insert, update, delete**) або оператора DDL. Тому існують дві форми тригерів:

Тригери DML;

Тригери DDL

2.2.Тригери можливо створювати за допомогою мови програмування C# або Visual Basic. У SQL Server існує два параметри, що визначають поведінку тригерів:

2.3. **AFTER.** Тригер виконується після успішного виконання викликали його команд. Якщо ж команди з якої-небудь причини не можуть бути успішно завершені, тригер не виконується.

2.4. Слід зазначити, що зміни даних в результаті виконання запиту користувача і виконання тригера здійснюється в тілі однієї транзакції: якщо відбудеться відкат тригера, то будуть відхилені і зміни користувача. Можна визначити кілька **AFTER**-тригерів для кожної операції (INSERT, UPDATE, DELETE).

2.5. Якщо для таблиці передбачено виконання кількох AFTER-тригерів, то за допомогою системної збереженої процедури «**SP_SETTRIGGERORDER»** можна вказати, який з них буде виконуватися першим, а який останнім. За умовчанням в SQL Server всі тригери є AFTER-тригерами.

2.6. **INSTEAD OF.** Тригер викликається замість виконання команд. На відміну від AFTER-тригера INSTEAD OF-тригер може бути визначений як для таблиці, так і для перегляду.

2.7. Для кожної операції **INSERT, UPDATE, DELETE** можна визначити тільки один INSTEAD OF-тригер.

2.8. Приклад використання тригера з оператором INSERT для таблиці **sotrydnik** наведений:

```
create TRIGGER prim1
ON sotrydnik
after INSERT
AS
DECLARE @oklad int
 select @oklad =(select oklad from inserted)
 if (@oklad)<2000
BEGIN
    ROLLBACK TRANSACTION
end
if not (@oklad)<2000
BEGIN
select @oklad=(select oklad from inserted)
end;
```

Завдання на виконання практичної роботи

Завдання 1. Створити процедуру щодо здійснює вибірку товару з таблиці ТОВАРИ.

Перевірити її роботу.

Завдання 2 Створити процедуру щодо здійснює вибірку даних по окремому товару з таблиці ТОВАРИ.

Перевірити її роботу.

Завдання 3. Розробити тригер, який виконується коли в таблицю Sales вставляється рядок або виконується її модифікація. Якщо дата замовлення не знаходиться в межах перших 15 днів місяця, рядок в таблицю не вводиться

Перевірити роботу даного тригера.

Завдання 4. Розробити тригер, який намагається видалити рядок з таблиці Stores. Якщо інформація стосується продажів, то тригер перешкоджає виконанню цього запиту.

Перевірити роботу тригера

Завдання 5. Створити тригер, який при видаленні роботи перевірятиме, призначені чи на неї співробітники чи ні. Якщо призначені, то робота не буде віддалятися.

Контрольні питання:

1. Поняття процедури та функції, призначення, властивості.
2. Переваги процедури у порівнянні з іншими об’єктами бази (запитами, представленнями).
3. Поняття локальної та глобальної процедури.
4. Призначення тригеру, типи тригерів.
5. Команди щодо створення процедури, вхідні та вихідні параметри.
6. Команди СКБД щодо перевірки стану роботи тригерів та їх заборони.
7. Тригери бази даних. Призначення, особливості застосування.

МЕТОДИЧНЕ ЗАБЕЗПЕЧЕННЯ ЗАНЯТТЬ

1. Електронні та друковані інформаційні ресурси.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Базова

1. Проектування баз даних у середовищі MS ACCESS 2010. Навчальний посібник / О. М. Верес, І. В. Рішняк. Львів : Видавництво Львівської політехніки, 2016. - 232 с.
2. Бекаревич, Ю. Б. Самоучитель Access 2016 / Ю. Б. Бекаревич, Н. В. Пушкина. — СПб.: БХВ-Петербург, 2016. — 432 с.:
3. Новиков Б. А. Основы технологий баз данных: учеб. пособие / Б. А. Новиков, Е. А. Горшкова; под ред. Е. В. Рогова. — М.: ДМК Пресс, 2019. — 240 с.
4. Мулеса О.Ю. Інформаційні системи та реляційні бази даних. Навч.посібник. – Електронне видання, 2018. – 118 с.
5. Советов, Б. Я. Базы данных : учебник для прикладного бакалавриата / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. — 2-е изд. — М. : Издательство Юрайт, 2015. — 463 с. — Серия : Бакалавр. Прикладной курс.
6. Гурвиц Г. MS Access 2013. Разработка приложений на реальних примерах / Г. Гурвиц. - СПб. : Питер, 2014. - 497 с.
7. Аблязов В.И . Проектирование баз данных в среде Microsoft Office Access 2003, 2007 и 2010./Аблязов В.И, Издательство Политехнического университета, - 2014,- 107 с.
8. Черняк О. І. Інтелектуальний аналіз даних : підручник / О. І. Черняк, П. В. Захарченко; Київський національний університет ім. Тараса Шевченка. - К. : Знання, 2014. - 599 с.
9. Л.С. Глоба, М.Ю. Терновой, Р.Л. Новогрудська, О.С. Штогрина /СТВОРЕННЯ ТА ОБРОБКА БАЗ ДАНИХ// Навчальний посібник для студентів технічних спеціальностей вищих навчальних закладів - Національний технічний університет України “Київський політехнічний інститут” Інститут телекомунікаційних систем – Київ -2013 456 С.

10. Литвин В.В. Бази знань інтелектуальних систем підтримки прийняття рішень. Монографія. Львів: Видавництво Львівської політехніки, 2011. - 240 с

11. Литвин В. В. Бази знань інтелектуальних систем підтримки прийняття рішень. Монографія. Львів: Видавництво Львівської політехніки, 2011. - 240 с.

12. Жежич П. І. Консолідовані інформаційні ресурси баз даних та знань : навчальний посібник / П. І. Жежич ; ред. В. В. Пасічник. - Львів : Вид-во Львівська політехніка, 2010. - 212 с.

13. Федько В. В. Створення баз даних та застосувань професійного спрямування. Лабораторний практикум: Навч.-практ. посібн. / Укл. В. В. Федько, В. І. Плоткін. – Харків : Вид. ХНЕУ, 2009.– 208 с.

14. Басюк Т. М. Основи інформаційних технологій : навчальний посібник / Т.М. Басюк Н.О. Думанський О.В. Пасічник ; За ред. В.В. Пасічника. - Львів : Новий світ, 2010. - 390 с.

15. Берко А. Ю. Система баз даних та знань. Кн. 1. Організація баз даних та знань : навчальний посібник / А.Ю. Берко, О.М. Верес, В.В. Пасічник ; За ред. В.В. Пасічника. - Львів : Магнолія, 2008. - 456 с.

16. Пасічник В. В. Сховища даних : навчальний посібник / В. В. Пасічник, Н.Б. Шаховська ; За ред. В.В. Пасічника. - Львів : Магнолія 2006, 2008. - 496 с. -

17. Гайна, Г. А. Основи проектування баз даних : навчальний посібник / Г. А. Гайна. - К. : Кондор, 2008. - 200 с.

18. Тейлор А. SQL для "чайников" / А. Тейлор ; пер. анг. С. А. Храмова. - 6-е изд. - М. : ООО "И.Д. Вильямс", 2008. - 352 с.

19. В.В. Пасічник В.А. Резніченко. Підручник для студентів ВНЗ з дисципліни «Організація баз даних та знань». – Київ, Видавнича група BHV, 2006.-384с.

20. Хендерсон К. Профессиональное руководство по SQL Server: хранимые процедуры, XML, HTML / К. Хендерсон. - СПб. : Питер, 2005. - 620 с.

21. Гайджарги В. І. Основи проектування та використання баз даних : навчальний посібник / В.І. Гайджарги, О.А. Дацюк. - К. : ІВЦ " Видавництво «Політехніка»", 2004. - 256 с.

Допоміжна

1. Персональная база данных для менеджера: Учебное пособие Автор: Князева М. Д. Жанр: MS Access Издательство: Форум 2014-224 С.

2. Разработка баз данных в Microsoft Access 2010. /С.В. Одиночкина, Санкт-Петербург: НИУ ИТМО, 2012, - 83 с

3. Видеосамоучитель. Microsoft Access/А. Днепров, Питер, 2010, - 240 с.

4. Д. Петкович Microsoft SQL Server . Руководство для начинающих. Пер. с анг - СПб.:БХВ-Петербург, 2009 – 752С.

5. Администрирование Microsoft SQL Server [Электронный ресурс] : учебный курс MCSA, MCSE, MCDBA. - М. : Издательско-торговый дом "Русская Редакция" ; СПб. : Питер, 2006.

6. Биков І. Ю. Microsoft Office в задачах економіки та управління : навчальний посібник / І.Ю. Биков, М.В. Жирнов, І.М. Худякова. - К. : ВД "Професіонал", 2006. - 264 с.

7. Проектирование и реализация баз данных Microsoft SQL Server [Электронный ресурс] : учебный курс MCSA, MCSE, MCDBA. - 3-е изд. - М. : Издательско-торговый дом "Русская Редакция" ; СПб. : Питер, 2006. - Пер. с англ.

8. Фленов М.Е. Transact-SQL- СПб.:БХВ-Петербург, 2006 – 576 С.

9. Профессиональное руководство по SQL Server: хранимые процедуры XML, HTML [Электронный ресурс]. - Электрон. текстовые дан. - СПб. : Питер, 2005.