МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

МЕТОДИЧНІ ВКАЗІВКИ

до практичних робіт з дисципліни

«ОРГАНІЗАЦІЯ БАЗ ДАНИХ»

(для здобувачів вищої освіти усіх форм навчання за спеціальностями "Комп'ютерні науки", "Комп'ютерна інженерія")

(електронне видання)

ЗАТВЕРДЖЕНО
на засіданні кафедри комп'ютерних наук
та інженерії
Протокол № 6 від 03.04.2026

Київ 2026

УДК 004.65

Методичні вказівки до практичних робіт з дисципліни «Організація баз даних» для здобувачів вищої освіти усіх форм навчання за спеціальностями «Комп’ютерні науки», «Комп’ютерна інженерія» / Уклад.: Шумова Л.О. – Київ : Вид-во СНУ ім. В. Даля, 2026. – 58 с.

Укладено на основі ОП підготовки бакалаврів за спеціальностями «Комп’ютерні науки», «Комп’ютерна інженерія» та робочої навчальної програми з дисципліни «Організація баз даних».

Теми практичних занять присвячені ключовим завданням, які виникають при проектуванні баз даних. Виконання робіт дозволяє отримати необхідні базові знання методів і способів проектування, отримати навички дослідження та використання алгоритмів і програмних засобів для систем управління базами даних, вивчити мову структурованих запитів SQL, питання обробки та безпеки баз даних.

Укладач: Л.О.Шумова, к.т.н., доцент

Рецензент: Є.В. Щербаков, к.т.н., доцент

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

КП – клас приналежності

НФ – нормальна форма

НФБК – нормальна форма Бойса –Кодда

РБД – реляційна база даних

СКБД – система керування базою даних, або Система управління базами даних (СУБД), англ. Database Management System (DBMS)

ВСТУП

Дисципліна «Організація баз даних» викладається для студентів денної і заочної форм навчання за спеціальностями 122 «Комп’ютерні науки», 123 «Комп’ютерна інженерія» протягом 6 семестра.

Метою практичних занять з дисципліни є: надання студентам поглиблених знань про реляційні бази даних; навчання концептуальному і логічному проектуванню баз даних для різних предметних областей з використанням алгоритму нормалізації відношень і за допомогою моделі «сутність-зв'язок»; навчання роботі зі стандартною мовою запитів реляційних баз даних SQL; навчання побудові фізичних моделей, маніпулюванню даними та створенню запитів і звітів за допомогою реляційної СКБД, зокрема MySQL.

Вивчення дисципліни «Організація баз даних» базується на таких дисциплінах як програмування, системне програмування, дискретна математика, тощо.

Основна увага приділяється теоретичним основам класичної та сучасної теорії, а також практичнім аспектам організації баз даних. Докладно розглянуто реляційну модель даних, реляційну алгебру Кодда, реляційне числення та теорію проектування баз даних. Значну увагу приділено мові SQL, а також методиці побудови різнотипних SQL запитів. Викладено основи фізичної організації баз даних.

При вивченні даної дисципліну формуються наступні компетентності:

– компетентності з інформаційних і комунікаційних технологій на підґрунті вивчення матеріалу курсу, роботою з центром електронних навчальних матеріалів університету;

– використання сучасних інструментальних систем для розробки програмного забезпечення.

Мета даних методичних вказівок – надати рекомендації по вирішенню домашніх і аудиторних задач, вказати літературні джерела, необхідні під час підготовки до занять.

1 ПРАКТИЧНЕ ЗАНЯТТЯ № 1

1.1 Тема заняття

Реляційна алгебра відношень

1.2 Мета заняття

Закріплення теоретичних знань про реляційний підхід, реляційну алгебру і придбання навичок у виконанні операцій реляційної алгебри.

1.3 Задачі заняття

Застосування теоретико-множинних операцій в опрацюванні даних. Виконання операцій реляційної алгебри

1.4 Постановка завдання

Для заданих відношень побудувати відношення, отримане в результаті виконання операцій реляційної алгебри над вихідними відношенями.

Для виконання практичного заняття необхідно користуватися матеріалами лекцій і літературними джерелами, наведеними у списку літератури. Основні поняття: база даних, , реляційна алгебра, реляційні відношення, схема відношень, кортеж, атрибут. Необхідно звернути увагу на такі поняття як однакова і різна схема відносин.

1.5 Основні відомості з теорії. Основи реляційної алгебри

Маніпулювання реляційними даними можливо за допомогою операторів реляційної алгебри. Реляційна алгебра являє собою набір операторів, що використовують відношення як аргументи, і повертають відношення в якості результату.

Перша версія цієї алгебри була визначена Едгаром Коддом [1], відповідно до якої визначають вісім реляційних операторів, об'єднаних у дві групи: теоретико-множинні оператори (об'єднання, перетин, різниця, декартовий добуток) і спеціальні реляційні оператори: (вибірка, проекція, з'єднання або конкатенація, обмеження), що є основними діями реляційної алгебри, які дозволяють створювати нові відношення з використанням вже наявних. Оператори

з'єднання, такі як декартів добуток, об'єднання та різниця, використовуються для створення нових відношень, що містять всі кортежі, які є частиною будь-якого з відповідних відношень. Оператори обмеження, такі як вибірка, використовуються для отримання нових відношень, що містять кортежі, які задовольняють заданій умові. Ці оператори є фундаментальними для розробки систем реляційних баз даних і є ключовими для виконання багатьох базових дій, таких як вибірка, проекція, декартів добуток, об'єднання та різниця. Відношення в реляційній моделі даних складаються з заголовка (*схеми*) і тіла (*кортежів*).

Відношення сумісні за типом, якщо вони мають ідентичні заголовки:

– відношення мають одну і туж множину імен атрибутів, тобто для будь-якого атрибута в одному відношенні знайдеться атрибут з таким же найменуванням в іншому;

– атрибути з однаковими іменами визначені на одних і тих же доменах.

Об'єднанням (*union*) двох сумісних по типу відношень R_1 і R_2 зі схемами $R_1(L)$ і $R_2(L)$, *де* L – певна множина атрибутів, (позначається як $(R_1 \cup R_2)$ називається таке реляційне відношення R зі схемою $R(L)$, що містить кортежі обох поєднуваних відношень, але без повторень (рис. 1.1):

$$R(L)=R_1(L)\cup R_2(L)=\{r \mid r\in R_1 \vee r\in R_2\}.$$

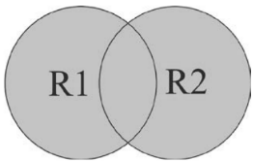


Рисунок 1.1 – Операція об’єднання ($R_1 \cup R_2$)

Приклад 1:

R_1

A	B
a_1	b_1
a_1	b_2
a_2	b_3

R_2

A	B
a_1	b_1
a_2	b_1

$R = R_1 \cup R_2$

A	B
a_1	b_1
a_1	b_2
a_2	b_1
a_2	b_3

Перетином (*intersection*) двох сумісних реляційних відношень R_1 і R_2 зі схемами $R_1(L)$ і $R_2(L)$, *де* L – певна множина атрибутів, (позначається як $R_1 \cap R_2$) називається таке реляційне відношення R зі схемою $R(L)$, яке містить кортежі, що входять до складу обох операндів (рис. 1.2):

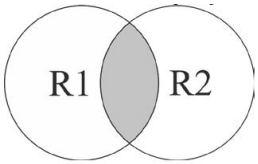


Рисунок 1.2 – Операція перетину ($R_1 \cap R_2$

Приклад 2:

R_1

A	B
a_1	b_1
a_1	b_2
a_2	b_3

R_2

A	B
a_1	b_1
a_2	b_1

$R_1 \cap R_2$

A	B
a_1	b_1

Різницею (*set difference*) двох сумісних реляційних відношень R_1 і R_2 зі схемами $R_1(L)$ і $R_2(L)$, *де* L – певна множина атрибутів, (позначається як R_1-R_2 або $R_1 \backslash R_2$) називається реляційне відношення R зі схемою $R(L)$, що містить ті кортежі з першого операнда R_1, яких немає у другому операнді R_2 (рис. 1.3):

$$R(L) = R_1(L) - R_2(L) = \{r \mid r \in R_1 \ \& \ r \notin R_2\}.$$

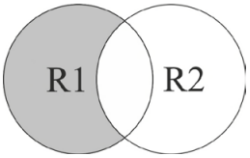


Рисунок 1.3 – Операція різниці (R_1 - R_2)

Приклад 3:

R_1

A	B
a_1	b_1
a_1	b_2
a_2	b_3

R_2

A	B
a_1	b_1
a_2	b_1

R_1-R_2

A	B
a_1	b_2
a_2	b_3

Зауважимо, що $R \cap S = R-(R-S)$.

Декартів добуток (*cartesian product*) реляційних відношень R і S зі схемами $R(A_1, A_2, \ldots A_n)$ та $S(B_1, B_2, \ldots B_m)$ відповідно, що позначається $R \times S$ *або* $R \otimes S$, називається відношення Q зі схемою $Q(A_1, A_2, \ldots A_n, B_1, B_2, \ldots B_m)$, яке містить усі можливі з'єднання кортежів відношення R з кортежами відношення S:

$$Q = R \times S = \{(r, s) \mid r \in R \ \& \ s \in S\}.$$

Приклад 4:

R_1

A	B
a_1	b_1
a_1	b_2
a_2	b_3

R_2

C	D
c_1	d_1
c_2	d_1

$R = R_1 \times R_2$

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_2	d_1
a_1	b_2	c_1	d_1
a_1	b_2	c_2	d_1
a_2	b_3	c_1	d_1
a_2	b_3	c_2	d_1

Операції вибірки й проекції є унарними, оскільки вони працюють із одним відношенням. Інші операції працюють із парами відношень, і тому їх називають бінарними операціями.

Вибірка (обмеження, селекція - *selection*). Спочатку дамо означення θ-порівнянності атрибутів. Нехай θ є одним з операторів порівняння: $=, \neq, \geq, >, \leq, <$ (набір операторів можна розширити). Атрибути A і B одного й того самого чи різних відношень називаються *θ-порівнянними*, якщо для будь-яких значень $a \in A$ і $b \in B$ результат операції $a\theta b$ є визначеним (істинним або хибним). Інакше кажучи, ця операція визначена на відповідних атрибутах. Набори атрибутів $L=(A_1,\ldots, A_k)$ та $M=(B_1,\ldots, B_n)$ називаються *θ-порівнянними*, якщо $k=n$ і A_i *θ-порівнянне* з B_i $(i=1,2,\ldots k)$. Тоді вираз $L\ \theta\ M$ розуміють так:

$$L\ \theta M\ =(A_1\ \theta B_1)\&\ldots\&\ (A_k\ \theta B_k).$$

Тепер дамо означення операції обмеження. Нехай L і M – набори *θ-порівнянних* атрибутів схеми відношення R. Тоді обмеженням реляційного відношення R за умовою $L\ \theta\ M$, що позначається $R[L\ \theta\ M]$, називається реляційне відношення, кортежі якого відповідають умові $L\ \theta\ M$:

$$S= R[L\ \theta M]=\{r \mid r\in R\ \&\ r[L]\ \theta r[M]\}$$

Множина M може складатися як з атрибутів, так і з констант.

Операція обмеження також записується як $\sigma_{L\ \theta\ M}(R)$.

Приклад 5:

R

A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_3	c_1
a_2	b_4	c_2

$R[A=a_2]$

A	B	C
a_2	b_3	c_1
a_2	b_4	c_2

Проєкцією (*projection*) реляційного відношення R зі схемою $R(A1,\ldots, Ak)$ за атрибутами $A_{i1},\ldots, A_{in}$, де $\{A_{i1},\ldots, A_{in}\}\subset\{A_1,\ldots, A_k\}$, що позначається $R[A_{i1},\ldots, A_{in}]$, називається таке відношення S зі схемою $S(A_{i1},\ldots, A_{in})$, кортежі якого отримані з кортежів відношення R шляхом видалення значень, що не належать атрибутам, за якими виконується проекція. При цьому в кінцевому відношенні повторні екземпляри кортежів видаляються. Операція проекції є фільтром по атрибутах, тобто деякий вертикальний фільтр.

Якщо r є кортежем відношення R, то записом $r[L]$, де L– підмножина атрибутів відношення R, позначимо множину тих елементів кортежу r, що відповідають значенням атрибутів з L. Тоді наведене вище визначення проекції може бути записане у такий спосіб:

$$S= R[A_{i1},\ldots, A_{in}]=\{\ r[A_{i1},\ldots, A_{in}] \mid r\in R\}.$$

Операція проекції також записується як $\pi_{Ai1,\ldots, Ain}(R)$.

Зауваження: за визначенням відношень усі дублюючі кортежі видаляються з результуючого

відношення.

Приклад 6:

R

A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_3	c_1
a_2	b_4	c_2

$R[A,C]$

A	C
a_1	c_1
a_2	c_1
a_2	c_2

Крім того, умова порівняння для θ-вибірки може містити довільне число простих порівнянь, з'єднаних логічними зв'язками ($AND(\wedge,\&)$,$OR(\vee)$). При необхідності можуть використовуватися круглі дужки/

Операція з'єднання (*join*) відношень, поряд з операціями вибірки і проекції, є однією з найбільш важливих реляційних операцій.

Зазвичай розглядається кілька різновидів операції з'єднання:

– загальна операція з'єднання;

– з'єднання (тета-з'єднання);

– екві-з'єднання;

– природне з'єднання.

Операції реляційної алгебри розглянуті в [2-6].

Є декілька типів запитів, які не можна виразити засобами реляційної алгебри. До них відносяться запити, що вимагають дати у відповіді список атрибутів, що задовольняють певним умовам; побудова транзитивного замикання відношень, побудова крос-таблиць. Для отримання відповідей на подібні запити доводиться використовувати процедурні розширення реляційних мов.

В реалізаціях конкретних реляційних СКБД зараз не використовується в чистому вигляді ні реляційна алгебра, ні реляційне числення. Фактичним стандартом доступу до реляційних даними стала мова SQL (Structured Query Language), яка детально розглянута далі.

1.6 Рішення типового завдання

Дано два відношення з однаковою схемою відношень (рис. 1.4): ***A*** і ***B***. Побудувати відношення ***R***, що є об'єднанням цих відношень:

$$\boldsymbol{R=A \cup B.}$$

відношення *A*:

Символ	Число
P	1
R	2
P	2
S	3
S	1

відношення *B*:

Символ	Число
R	2
S	3
F	1

Рисунок 1.4 – Вихідні відношення

Відношення ***R=A*** ∪ B:

Символ	Число
P	1
R	2
P	2
S	3
S	1
F	1

1.7 Завдання до самостійної роботи

1.Побудувати відношення: *R1=R*∪ *A*, *R2=R1*∪ *B*.

2. Побудувати відношення: *R3 = A*∩*C*∩*B*, *R4 = R1*∩*C R2* (рис. 1.5)

3. Побудувати відношення: *R5=A\B, R6=B\A, R7=R1\A*

4. Дано два відношення з різною схемою даних: *C* і *D* (рис.1.5). Побудувати відношення *S*, що є декартовим добутком цих відношень: *S=C* ⊗ *D.*

Відношення *C*

№	Прізвище
1	Коваль
2	Мельник
3	Польовий

Відношення *D*

№	Им’я
1	Василій
2	Петро
3	Миколай

Рисунок 1.5 – Вихідні відношення *C* і *D*

6. Дано відношення *E*. (рис.1.6). Побудувати відношення *P*, яке є результатом вибору за умовою *P=E [рік народження < 1980]*

Відношення E:

№	Прізвище	Рік народження	Місто
1	Коваль	1970	Харків
2	Мельник	1972	Харків
3	Польовий	1979	Львів
4	Коваль	1982	Київ
5	Самчук	1970	Рівне

Рисунок 1.6 – Вихідне відношення *E*

7. Побудувати відношення:

P1=E[Рік народження >= 1972],

P2=E[Місто = ‘Харків’],

P3=E[№ >= 3].

1.8 Контрольні питання

1. У чому полягає сенс операції об'єднання, віднімання та перетину?

2. Які операції реляційної алгебри є базовими і чому?

3. Що таке схема даних і в чому відмінність між однаковою і різною схемами? Наведіть приклад.

4. Яким чином будуть відрізнятися результати реляційних операцій для однаковою і різних схем даних?

5. У чому відмінність між операціями селекції та проекції?

1.9 Рекомендована література та інші джерела інформації

1. The Database Relational Model: A Retrospective Review and Analysis: A Historical Account and Assessment of E. F. Codd's Contribution to the Field of Database Technology https://archive.org/details/databaserelation00date

2. Конспект лекцій з дисципліни «Організація баз даних» (для студентів 3 курсу денної та заочної форми навчання за спеціальностями 122 «Комп’ютерні науки», 123 «Комп’ютерна інженерія») / Уклад.: Сафонова С.О. – Сєвєродонецьк: вид-во СНУ ім. В. Даля, 2018. - 166 с.

3. Мартиненко Г. Ю. Концептуальне та логічне проектування реляційних баз даних [Електронний ресурс] : навч.-метод. посібник / Г. Ю. Мартиненко ; Нац. техн. ун-т "Харків. політехн. ін-т". – Електрон. текст. дані. – Харків, 2023. – 91 с. https://repository.kpi.kharkov.ua/handle/KhPI-Press/70293

4. Реляційна алгебра. Реляційне числення [Електронний ресурс] — Режим доступу : https://evnuir.vnu.edu.ua/bitstream/123456789/18857/1/Relyatsiyna_alhebra.pdf

5. Реляційна алгебра [Електронний ресурс] — Режим доступу : https://rdb.dp.ua/uk/chapter_05

6. Вікіпедія – Вільна енциклопедія – *https://uk.wikipedia.org/wiki/ Реляційна_алгебра*

2 ПРАКТИЧНЕ ЗАНЯТТЯ № 2

2.1 Тема заняття

Нормалізація схем відношень

2.2 Мета заняття

Закріплення теоретичних знань з процесу нормалізації відношень та набуття навичок у приведенні схем відношень до першої, другої і третьої нормальної форми

2.3 Задачі заняття

Застосування декомпозиції відношень в процесах проектування.

При підготовці до заняття необхідно користуватися матеріалами лекцій та джерелами, наведеними у переліку пропонованої літератури.

При вивченні методів приведення відношень до другої і третьої нормальних форм необхідно звернути увагу на значення семантичного аналізу предметної області для виявлення функціональних і транзитивних залежностей.

2.4 Постановка завдання

1. Вибрати будь-яку предметну область (див. Варіанти завдань, п. 2.7) і виділити в ній будь-який об'єкт, який має багато властивостей, важливих для розгляду.

2. Зіставити властивостям об'єкта атрибути, а класу об'єктів - відношення.

3. Визначити ключовий атрибут і атрибути, які мають складовий характер (наприклад атрибут "Діти" для відношення "Співробітник" мають такі значення як ім'я дитини, рік її народження тощо, і їх може бути декілька, включаючи пусте значення).

4. Застосувати до отриманого відношення процедуру приведення до першої нормальної форми.

5. Визначити атрибути, які функціонально повно залежать від складеного ключа.

6. Застосувати до відношення, отриманого в п.4, процедуру приведення до другої нормальної форми.

7. Для відношень, отриманих в п.6, визначити наявність транзитивної залежності і при потребі застосувати процедуру приведення до третьої нормальної форми.

2.5 Основні відомості з теорії

Нормалізація – процес побудови баз даних за спеціальною методикою, що передбачає дотримання певного набору норм і правил, вперше запропонований Е. Коддом з метою усунення можливих недоліків проектування та аномалій баз даних. В основі методів нормалізації лежить поняття нормальної форми.

Нормальна форма *(НФ)* – це спосіб побудови відношення в якому через виконання певної множини вимог (норм) воно набуває множини наперед визначених властивостей. Нормальна форма визначається за принципом "якщо – то". Нормалізація за Коддом – процесс поетапного переходу від однієї нормальної форми до іншої шляхом виконання певних вимог, кожна з яких доповнює попередні і надає відношенню додаткових якостей.

Перша нормальна форма – *1НФ (First Normal Form – 1NF*). Вимогою першої нормальної форми є зображення даних *у вигляді відношень*, побудованих на основі *іменованих атрибутів* та їх значень, з дотриманням усіх вимог, які визначаються реляційною моделлю:

−відсутність повторень атрибутів та кортежів;

−відсутність впорядкування кортежів та атрибутів;

−елементарність значень атрибутів у кортежах відношення.

Друга нормальна форма – *2НФ (Second Normal Form – 2NF).* Відношення знаходиться у другій нормальній формі, якщо воно задовольняє вимогам першої нормальної форми, і в ньому *визначений ключ*, від якого повністю функціонально залежить кожен непервинний атрибут.

Третя нормальна форма – *3НФ (Third Normal Form – 3NF).* Відношення задовольняє вимогам третьої нормальної форми, якщо воно знаходиться в другій нормальній формі і *для кожного непервинного атрибута* підтримується *повна і нетранзитивна* функціональна залежність від єдиного ключа.

Нормальна форма Бойса –Кодда – *НФБК (Boyce-Codd Normal Form – BCNF).* Відношення задовольняє нормальної форми Бойса-Кодда, якщо воно знаходиться в другій нормальній формі і *для кожного атрибут* підтримується *повна і нетранзитивна* функціональна залежність від єдиного ключа.

Вважається, що база даних задовольняє вимогам певної нормальної форми, якщо всі відношення, що входять до неї, задовольняють вимогам цієї *НФ*, отже, якщо в базі даних присутні відношення у *2НФ* і *НФБК*, то в загальному вона нормалізована до другої нормальної форми.

Процес приведення відношення до *НФБК* покладено в основу алгоритму декомпозиції. Його основні етапи:

−створення універсального відношення, в яке включають всі необхідні атрибути. Це відношення має знаходиться в *1НФ*;

−визначення всіх функціональних залежностей між атрибутами відношення;

−перевірка відношення на приналежність *НФБК*. Якщо відношення не знаходиться в НФБК, то проводиться нормалізація, тобто розбиття відношення на 2. В окреме відношення виділяються ті атрибути, які заважають відношенню перебувати в *НФБК*. Для отриманих відношень виконуються п.п. 2 і 3 до тих пір, поки всі відношення не будуть знаходитися в НФБК.

Існують рівні нормалізації вище *НФБК*, але на практиці вони застосовуються досить рідко, так як сильно ускладнюють розробку структур даних і знижують їх функціональність.

Метод декомпозиції, або приведення відносин до *НФБК*, ефективний тільки в тому випадку, якщо кількість атрибутів відношення не перевищує 15-20. Якщо ж необхідно зберігати в базі даних значно більший обсяг інформації, то краще скористатися іншими методами.

2.6 Рішення типового завдання

Початкове відношення:

ФІЛЬМ (Назва фільму, жанр, рік випуску, ПІБ режисера, дата народження, кількість фільмів).

Виконати перевірку цього відношення на приналежність НФБК.

Визначимо наступні **функціональні залежності** цього відношення:

Назва фільму → жанр

Назва фільму → рік випуску

Назва фільму → ПІБ режисера

ПІБ режисера → дата народження

ПІБ режисера → кількість фільмів

Детермінанти відносини: *Назва фільму*, *ПІБ режисера*.

Можливий ключ: *Назва фільму*.

Детермінант *ПІБ режисера* не є можливим ключем, отже, відношення ФІЛЬМ не знаходиться в НФБК і його потрібно нормалізувати. У нове відношення слід виділити атрибути, які залежать від ПІБ режисера. Отримаємо 2 відносини:

ФІЛЬМ (Назва фільму, жанр, рік випуску, ПІБ режисера)

РЕЖИСЕР (ПІБ режисера, дата народження, кількість фільмів).

Нескладно переконатися, що отримані відносини знаходяться в *НФБК*.

2.7 Завдання до самостійної роботи

Варіанти завдань:

1. Установи. Об'єкти: співробітник, відділ.
2. ВУЗ. Об'єкти: студент, група.
3. Бібліотека. Об'єкти: розділ, книга.
4. Рух вантажів. Об'єкти: відправник, вантаж.
5. Приймальна комісія. Об'єкти: абітурієнт, факультет.
6. Аеропорт. Об'єкти: рейс, пасажир.
7. Транспортні засоби. Об'єкти: власник, автомобіль.
8. Продукція. Об'єкти: виробник, виріб.
9. Навчання. Об'єкти: предмет, викладач.
10. Спорт. Об'єкти: команда, гравець.
11. Ремонт будівель. Об'єкти: підрядник, будівля.
12. Живопис. Об'єкти: музей, картина.
13. Торгівля. Об'єкти: магазин, продавець.
14. Ветеринарна служба. Об'єкти: власник, тварина.
15. Звукозапис. Об'єкти: альбом, виконавець.
16. Література. Об'єкти: автор, твір.
17. Торгівля. Об'єкти: відділ, товар.

2.8 Контрольні питання

1. Сформулюйте визначення терміну *відношення*.

2. У чому полягає процедура приведення *відношення* до *першої нормальної форми*?

3. Що таке функціональна залежність і які методи її виявлення?

4. У чому полягає процедура приведення *відношення* до другої нормальної форми?

5. Що таке *транзитивна залежність* і в чому полягає процедура приведення *відношення* до третьої нормальної форми?

2.9 Рекомендована література та інші джерела інформації

1. Конспект лекцій з дисципліни «Організація баз даних» (для студентів 3 курсу денної та заочної форми навчання за спеціальностями 122 «Комп’ютерні науки», 123 «Комп’ютерна інженерія») / Уклад.: Сафонова С.О. – Сєвєродонецьк: вид-во СНУ ім. В. Даля, 2018. - 166 с.

2. Демиденко М.А. Введення в сучасні бази даних : навч. посіб. / М.А. Демиденко; НТУ «Дніпровська політехніка». – Д. : 2020. – 38 с.

3. Вікіпедія – Вільна енциклопедія – https://uk.wikipedia.org/wiki/Реляційна_модель_даних

4. Вікіпедія – Вільна енциклопедія – https://uk.wikipedia.org/wiki/Реляційна_база_даних

5. Вікіпедія – Вільна енциклопедія – https://uk.wikipedia.org/wiki/Нормалізація_баз_даних

3 ПРАКТИЧНЕ ЗАНЯТТЯ № 3

3.1 Тема заняття

Проектування БД методом «сутність-зв'язок»

3.2 Мета заняття

Закріплення теоретичних знань про методи проектування і придбання навичок в проектуванні БД методом «сутність-зв'язок».

3.3 Задачі заняття

Вивчити:

- призначення та зміст концептуальної моделі даних;
- методи та засоби визначення предметної області БД;
- принципи побудови моделі "сутність – зв'язок";
- зміст та призначення понять "сутність", " атрибут", "зв'язок";
- порядок визначення класів та примірників сутностей;
- порядок визначення атрибутів сутностей;
- призначення та зміст зв'язків між сутностями.

Виконати проектування БД методом «сутність-зв'язок».

Для виконання практичної роботи необхідно користуватися матеріалами лекцій, і наведеними літературними джерелами (п. 3.9). Основні поняття: база даних, сутність, первинний ключ, зовнішній ключ, відношення, атрибути сутності, зв'язки між сутностями, діаграма ER-типу, діаграма ER-примірників, схема відносин, кортеж. Необхідно звернути увагу на такі поняття як однакова і різна схема відносин.

3.4 Постановка завдання

Виконати проектування БД методом «сутність-зв'язок» для створення інформаційної системи згідно обраної предметної області (п. 3,7).

3.5 Основні відомості з теорії

3.5.1 Правила формування відносин з діаграм ER-типу

Правила формування відносин враховують такі особливості діаграм ER-типу:

- види зв'язку між сутностями (1:1, 1: N, M: N);
- класи приналежності примірників сутностей (обов'язковий і необов'язковий).

Розглянемо формулювання шести правил формування відносин на основі діаграм ER-типу.

Правило 1. Якщо ступінь бінарної зв'язку 1:1 і клас приналежності обох сутностей обов'язковий, то формується одне відношення. Первинним ключем цього відношення може бути ключ будь-який з двох сутностей.

Правило 2. Якщо ступінь зв'язку 1:1 і клас приналежності (КП) однієї сутності обов'язковий, а другий - необов'язковий, то під кожну з сутностей формується по відношенню з первинними ключами, які є ключами відповідних сутностей. Далі до відношення, сутність якого має обов'язковий КП, додається в якості атрибута ключ сутності з необов'язковим КП.

Правило 3. Якщо ступінь зв'язку 1:1 і клас приналежності обох сутностей є необов'язковим, то необхідно використовувати три відносини. Два відносини відповідають пов'язують сутностей, а третє відношення складається для зв'язку двох сутностей. У відношення для зв'язку включаються ключі обох сутностей, а його первинним ключем можна вибрати ключ будь сутності.

Правило 4. Якщо ступінь зв'язку між сутностями 1: N (або N: 1) і клас приналежності N-зв'язковий сутності обов'язковий, то достатньо формування двох відносин (по одному на кожну з сутностей). При цьому первинними ключами цих відносин є ключі їх сутностей. Крім того, ключ 1-зв'язковий сутності додається як атрибут (зовнішній ключ) у відношення, відповідне N-зв'язковій сутності.

Правило 5. Якщо ступінь зв'язку 1:N (N:1) і клас приналежності N-зв'язковий суті є необов'язковим, необхідно формування трьох відносин. Два відношення відповідають сполучним сутностям, ключі яких є первинними у цих відносинах. Третє відношення є зв'язковим між першими двома. У нього входять ключі обох сутностей, а його первинним ключем буде ключ N-зв'язної сутності.

Правило 6. Якщо ступінь зв'язку M:N, то незалежно від класу приналежності сутностей формуються три відношення. Два відношення відповідають сполучним сутностям та їх ключі є первинними ключами цих відношень. Третє відношення є зв'язковим між першими двома, а його ключ об'єднує ключові атрибути що пов'язують відношення.

3.6 Рішення типового завдання

Завдання: виконати проектування бази даних методом «сутність-зв'язок» для створення інформаційної системи «Футбольний клуб».

Визначимо сутності, що представляють інтерес з точки зору проектування - «Футболіст», «Команда», «Тренер», «Тренувальна база», «Спонсор».

Для кожної сутності визначимо атрибути, які характеризують екземпляри сутностей, а також первинні ключі сутності.

Атрибути сутності «Команда»:

- Код команди;
- Назва команди;
- Місто;
- Розрахунковий рахунок;
- Ліга;
- Рік створення команди.

Первинним ключем сутності «Команда» є атрибут *Код команди*. Первинним ключем може бути і складний ключ з 2 атрибутів - *Назва команди* і *Місто*. Цей набір атрибутів називається можливим ключем.

Атрибути сутності «Футболіст»:

- Код гравця;
- ПІБ;
- Номер;
- Розряд;
- Дата народження.

Для однозначної ідентифікації футболіста недостатньо вказати ПІБ, тому що можливі однакові призвіща та ініціали. Ключем цієї сутності може бути номер паспорта або ідентифікаційний код. Для сутності «Футболіст» введемо атрибут «Код гравця», який буде первинним ключем цієї сутності.

Атрибути сутності «Тренер»:

- Код тренера;
- ФІО_Т;
- Адреса;
- Стаж роботи;
- Термін контракту.

Первинний ключ сутності - «Код тренера».

Атрибути сутності «Тренувальна база»:

- Назва бази;
- Кількість полів;
- Вартість оренди.

Первинний ключ сутності - «Назва бази».

Атрибути сутності «Спонсор»:

- Код спонсора;
- Найменування;
- Факс / телефон.

Первинний ключ сутності - «Код спонсора».

Наступний етап проектування - **створення ER-діаграми**. Для цього необхідно визначити зв'язки між сутностями і характеристики зв'язків. До таких характеристик відносяться ступеня зв'язків і класи приналежності сутностей. Ці параметри ER-діаграми є визначальними для отримання попередніх відносин.

Розглянемо зв'язок між сутностями.

Зв'язок «Футболіст-Команда». В одній команді грає багато футболістів. Якщо вважати, що дана БД містить відомості тільки про клубні командах (не розглядаються збірні), то гравець не може належати до декількох командах. Отже, зв'язок між сутностями «Футболіст» - «Команда» має ступінь зв'язку «один до багатьох» (1: N). Будемо вважати, що не може існувати команда в якій немає жодного гравця, а футболіст може не входити ні в одну команду. З цього випливає, що клас приналежності сутності «Команда» - обов'язковий, а сутності «Футболіст» - необов'язковий. На рис. 3.1, 3.2 представлені отримані ER-діаграми.

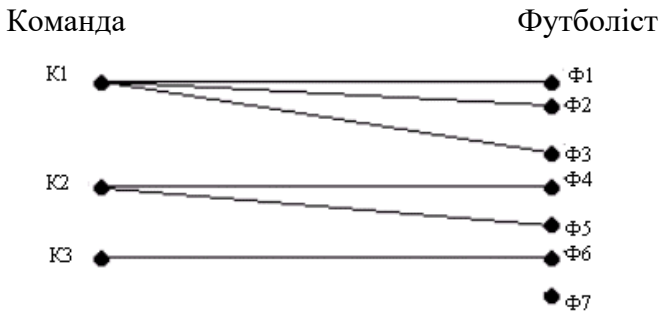


Рисунок 3.1 – Діаграма ER-примірників сутностей «Команда-Футболіст»

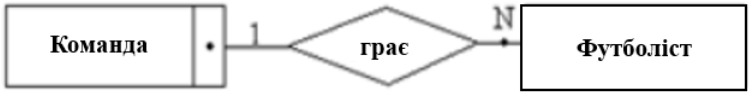


Рисунок 3.2 – Діаграма ER-типу сутностей «Команда-Футболіст»

Подібні обгрунтування вибору ступеня зв'язку і класів приналежності необхідно зробити для кожної пари пов'язаних сутностей.

Зв'язок «команда-тренер». З однією командою може працювати кілька тренерів, але тренер не працює з декількома командами. Отже, ступінь зв'язку між сутностями 1:N. З командою обов'язково працює тренер, значить, клас приналежності сутності «Команда» – обов'язковий. Тренер в даний момент може не працювати ні з якою командою, отже, клас приналежності сутності «Тренер» необов'язковий.

Зв'язок «команда-тренувальна база». Будемо вважати, що команда укладає договір з однією базою, але на одній тренувальній базі можуть одночасно готуватися кілька команд, тобто ступінь зв'язку один-до-багатьох (1: N).

База може знаходитися на реконструкції, ремонті і не приймати команд для тренувань. Тому встановлюємо для сутності «тренувальна база» необов'язковий клас приналежності. Команда обов'язково повинна мати базу для підготовки, клас приналежності сутності «Команда» - обов'язковий.

Зв'язок «команда-спонсор» припускає, що команда може мати кількох спонсорів, а спонсор може перерахувати гроші кільком командам.

Зв'язок між цими сутностями «багато-до-багатьох». Клас приналежності сутності «Спонсор» - необов'язковий, а сутності «Команда» - обов'язковий.

Отримана діаграма ER-типу представлена на рисунку 3.3.

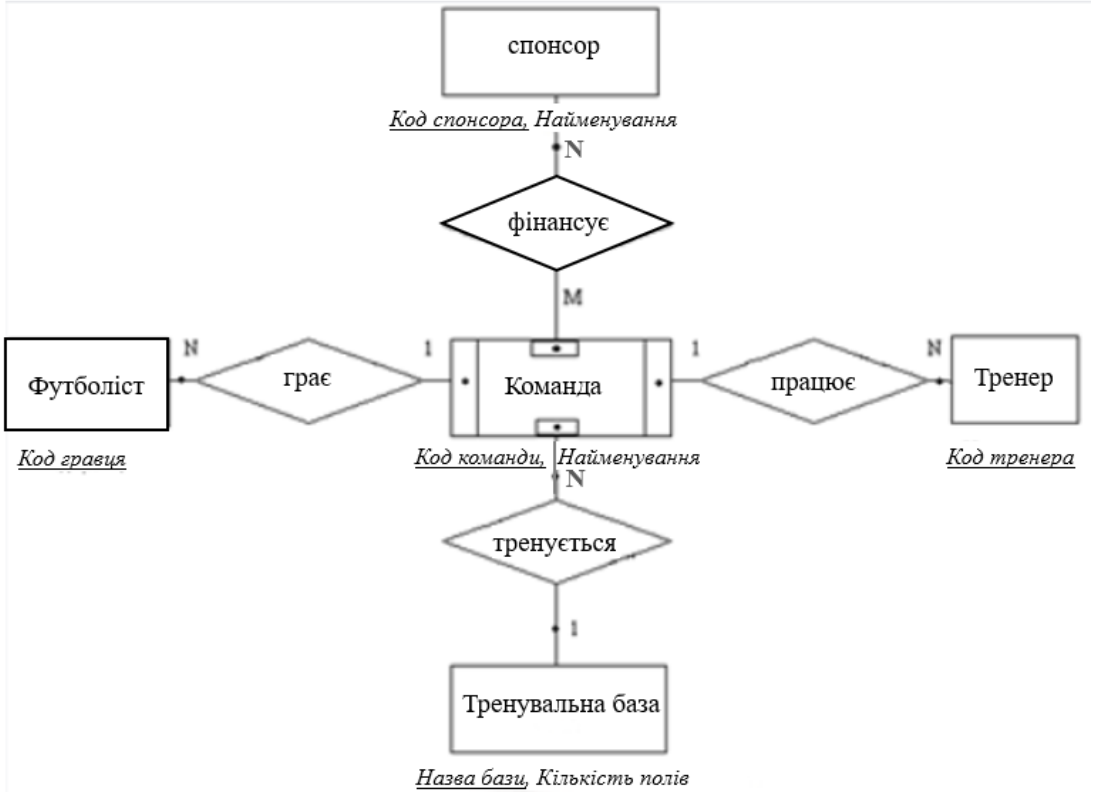


Рисунок 3.3 – Діаграма сутностей та зв’зків БД «Футбольний клуб»

На основі діаграми ER-типу за допомогою шести сформульованих правил отримаємо попередні відношення.

Зв'язок «Футболіст-Команда». Відповідно до правила 5 отримуємо 3 відносини:

КОМАНДА (Код команди, Назва команди, Місто, Розрахунковий рахунок, Ліга, Рік створення,)

ФУТБОЛІСТ (Код гравця, ПІБ, номер, дата народження, Розряд).

ГРАЄ (Код гравця, Код команди).

Зв'язок «Команда-Тренер». За правилом 5 повинні бути отримані 3 відносини, але відношення КОМАНДА вже створено при розгляді зв'язку «Футболіст-Команда», тому одержуємо відношення:

ТРЕНЕР (Код тренера, ФІО_Т, Адреса, Стаж роботи, Термін контракту).

ПРАЦЮЄ (Код тренера, Код команди).

Зв'язок «Команда-Тренувальна База». За правилом 4 потрібно отримати 2 відношення. Назва бази слід додати у відношення КОМАНДА, тому відношення коригується наступним чином:

КОМАНДА (Код команди, Назва команди, Місто, Розрахунковий рахунок, Ліга, Рік створення, Назва бази).

Друге відношення:

ТРЕНУВАЛЬНА БАЗА (Назва бази, Кількість полів, Вартість оренди).

Зв'язок «команда-спонсор». За правилом 6 слід створити 3 відносини. Крім відносини КОМАНДА, отримаємо відносини:

СПОНСОР (Код спонсора, Найменування, Телефон)

ФІНАНСУВАННЯ (Код спонсора, Код команди).

В результаті проектування отримано такі попередні відносини:

КОМАНДА (Код команди, Назва команди, Місто, Розрахунковий рахунок, Ліга, Рік створення, Назва бази);

ФУТБОЛІСТ (Код гравця, ПІБ, номер, дата народження, Розряд);

ГРАЄ (Код гравця, Код команди);

ТРЕНЕР (Код тренера, ФІО_Т, Адреса, Стаж роботи, Термін контракту);

ПРАЦЮЄ (Код тренера, Код команди);

ТРЕНУВАЛЬНА БАЗА (Назва бази, Кількість полів, Вартість оренди);

СПОНСОР (Код спонсора, Найменування, Телефон);

ФІНАНСУВАННЯ (Код спонсора, Код команди).

Отримані відношення перевіримо на приналежність нормальній формі Бойса-Кодда. Розглянемо цей процес на прикладі відношення КОМАНДА.

Визначаємо функціональні залежності цього відношення:

Код команди → Назва команди

Код команди → Місто

Код команди → Розрахунковий рахунок

Код команди → Ліга

Код команди → Рік створення

Код команди → Назва бази

Детермінантом всіх функціональних залежностей є атрибут Код команди.

Можливі ключі відносини:

- Код команди;
- Назва команди + Місто;
- Розрахунковий рахунок.

Так як детермінант є можливим ключем, робимо висновок про те, що ставлення знаходиться в НФБК і не вимагає нормалізації.

Аналогічні міркування можна застосувати і до інших відносин.

3.7 Завдання на самостійну роботу

Виконати проектування бази даних методом «сутність-зв'язок» для створення інформаційної системи згідно варіанту (кількість сутностей не менше 4). При цьому необхідно:

– описати предметну область, визначити об'єкти, які будуть включені в БД і визначити зв'язки між ними;

– створити ER-діаграми, обгрунтувати типи зв'язків між сутностями, вибрати необхідні правила і привести отримані попередні стосунки.

Варіанти завдань

1. Абітурієнти (робота приймальної комісії ВНЗ).
2. Автоперевезення.
3. Агентство з працевлаштування.
4. Адміністратор готелю.
5. Бібліотека.
6. Відділ кадрів підприємства.
7. Деканат.
8. Кафедра.
9. Музичний каталог.
10. Оптова база.Оренда приміщень.

11. Поліклініка.
12. Продаж автомобілів.
13. Рекламне агентство.
14. Розклад занять у ВНЗ.
15. Склад.
16. Станція техобслуговування.
17. Фірма зі збирання комп'ютерів.
18. Футбольний чемпіонат.

3.8 Контрольні питання

1. Які етапи проектування БД?
2. Що таке інфологічне моделювання? Що є результатом інфологічного моделювання?
3. Що таке даталогічне моделювання? Що є результатом даталогічного моделювання?
4. Як сутності ER-моделі перетворюються у відношення?
5. Перерахуйте три типи зв'язків? Наведіть приклади кожного типу.
6. Визначте термін зовнішній ключ і приведіть приклад?
7. Перерахуйте етапи проектування методом «сутність-зв'язок»?

3.9 Рекомендована література та інші джерела інформації

1. The Database Relational Model: A Retrospective Review and Analysis: A Historical Account and Assessment of E. F. Codd's Contribution to the Field of Database Technology https://archive.org/details/databaserelation00date
2. Конспект лекцій з дисципліни «Організація баз даних» (для студентів 3 курсу денної та заочної форми навчання за спеціальностями 122 «Комп’ютерні науки», 123 «Комп’ютерна інженерія») / Уклад.: Сафонова С.О. – Сєвєродонецьк: вид-во СНУ ім. В. Даля, 2018. - 166 с.
3. Мартиненко Г. Ю. Концептуальне та логічне проектування реляційних баз даних [Електронний ресурс] : навч.-метод. посібник / Г. Ю. Мартиненко ; Нац. техн. ун-т "Харків. політехн. ін-т". – Електрон. текст. дані. – Харків, 2023. – 91 с. https://repository.kpi.kharkov.ua/handle/KhPI-Press/70293
4. Демиденко М.А. Введення в сучасні бази даних : навч. посіб. / М.А. Демиденко; НТУ «Дніпровська політехніка». – Д. : 2020. – 38 с.
5. Вікіпедія – Вільна енциклопедія – *https://uk.wikipedia.org/wiki*

4 ПРАКТИЧНЕ ЗАНЯТТЯ № 4

4.1 Тема заняття

Мова запитів SQL – як універсальний інструмент доступу до БД. Мова визначення даних DDL.

4.2 Мета заняття

Закріплення теоретичних знань мови DDL як компонента SQL для створення БД.

4.3 Задачі заняття

Вивчення компонентів мови SQL.

Вивчення базових операторів мови DDL зі створення БД.

Виконання операцій зі створення таблиць БД з повним описом їх структури.

4.4 Постановка завдання

За наведеними прикладами засобами мови DDL створити таблиці БД з повним описом їх структури.

Для виконання практичного заняття необхідно користуватися матеріалами лекцій і літературними джерелами, що рекомендовані.

4.5 Основні відомості з теорії

4.5.1 Мова запитів SQL (Structured Query Language — структурована мова запитів) була створена фірмою IBM у рамках роботи з побудови системи управління реляційними базами даних на початку 70-х років. Американський національний інститут стандартів (ANSI) поклав цю мову в основу стандарту мов реляційних баз даних, прийнятого міжнародною організацією стан дартів (ISO).

Існує комерційний стандарт мови SQL, розроблений консорціумом виробників БД SQL Access Group. Ця група створила такий варіант мови, який використовується більшістю систем і дає змогу їм «розуміти» одна одну. Було розроблено стандартний інтерфейс мови CLI (Common Language Interface) для всіх основних варіантів мови SQL. Цей інтерфейс, формалізований фірмою Microsoft, дістав назву ODBC (Open Database Connectiviti — відкритий доступ до даних). ODBC — це інтерфейс доступу до даних, які зберігаються, під управлінням різних

СКБД. ODBC має цілий набір драйверів, за допомогою яких одна СКБД може працювати з даними інших систем. Архітектуру ODBC зображено на рис. 4.1 [1].

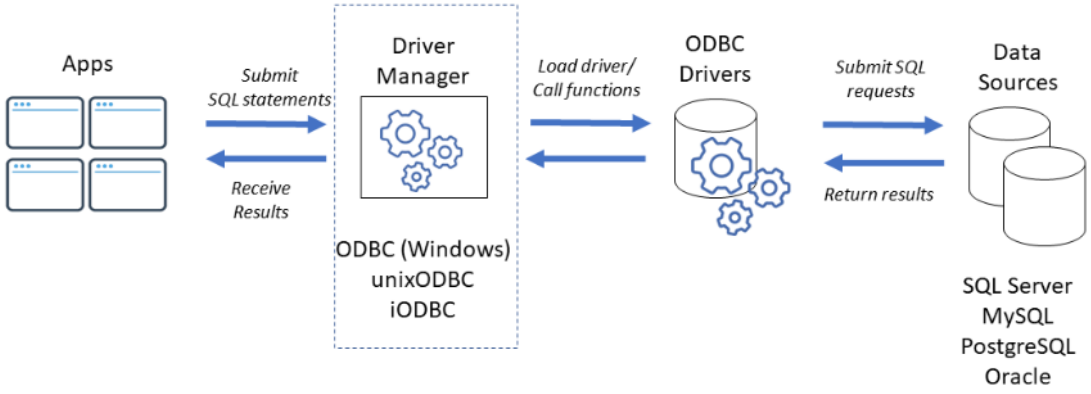


Рисунок 4.1 - Архітектура ODBC

В ідеалі, будь-яка мова роботи з БД повинна надавати користувачеві наступні можливості:

1) створювати БД і таблиці з повним описом їх структури;

2) виконувати основні операції маніпулювання даними, такі як вставка, модифікація й вилучення даних з таблиць;

3) виконувати прості та складні запити, здійснювати перетворення неопрацьованих даних у необхідну інформацію;

4) відповідати деякому визнаному стандарту, що дозволить використовувати той самий синтаксис і структуру команд при переході від однієї СКБД до іншої.

Крім того, мова роботи з БД повинна вирішувати всі зазначені вище завдання при мінімальних зусиллях з боку користувача, а структура й синтаксис його команд повинні бути досить прості й доступні для вивчення. Мова SQL задовольняє практично всім цим вимогам.

SQL можна в повній мірі віднести до традиційних мов програмування, вона не містить традиційні оператори, що керують ходом виконання програми, оператори опису типів і багато іншого, вона містить тільки набір стандартних операторів доступу до даних, що зберігаються в БД. Оператори SQL вбудовуються в базову мову програмування, якою може бути будь-яка стандартна мова типу C++ тощо. Крім того, оператори SQL можуть виконуватися безпосередньо в інтерактивному режимі.

На відміну від реляційної алгебри, де були представлені тільки операції запитів до БД, SQL є повною мовою, в ній присутні не тільки операції запитів, але і оператори, відповідні Data Definition Language (DDL) – мови опису даних. Крім того, мова містить оператори, призначені для управління (адміністрування) БД [2].

4.5.2 Оператори мови SQL можна умовно розділити на чотири підмови:

- мова визначення даних (Data Definition Language – DDL);
- мова маніпулювання даними (Data Manipulation Language – DML);
- мова управління даними (Data Control Language – DCL);
- мова управління транзакціями (Transaction Control Language – TCL).

Базові оператори мови SQL наведено в табл. 4.1 [3].

На практиці *для визначення структури БД використовується DDL-оператори*, а для заповнення цих відношень даними й вибірки з них інформації за допомогою запитів – *DML-оператори* [4-5].

Таблиця 4.1 – Базові оператори мови SQL

Підмова	Назва оператора	Призначення
DDL	CREATE	створення нових об'єктів бази даних, таких як таблиці, індекси тощо.
	ALTER	зміна структури вже існуючих об'єктів БД, наприклад, додавати, змінювати або видаляти колонки в таблиці
	DROP	видалення снуючих об'єктів БД, таких як таблиці, індекси тощо
	TRUNCATE	видалення всіх записів з таблиці, зберігаючи саму таблицю
	RENAME	зміна імені вже існуючих об'єктів БД, таких як таблиці, стовпці, індекси тощо
DML	SELECT	вибірка з вмісту таблиці
	INSERT	порядкове додавання даних в таблицю
	UPDATE	модифікація даних таблиці
	DELETE	видалення рядків таблиці
DCL	GRANT	надання прав користувачу
	DENY	явна заборона для користувача
	REVOKE	відміна заборони/дозволу користувачу
TCL	BEGIN TRANSACTION	початок транзакції
	COMMIT	збереження змін, внесений транзакцією
	ROLLBACK	відкат, повертає вміст таблиці БД в її початковий стан, зафіксований останньою командою COMMIT

4.5.3 Мова DDL – розділ SQL, який вміщує команди роботи зі структурою, структурними компонентами та обмеженнями цілісності СКБД. Дані команди не є транзакціями і не потребують COMMIT (примусового внесення даних в БД), бо автоматично виконуються і результат їхньої діяльності автоматично потрапляє до БД.. Ця мова використовується для опису структури БД, створення, зміни та видалення об'єктів БД (таблиць, індексів, схем тощо).

4.5.4 Команди для роботи з БД:

- перегляд доступних баз даних: SHOW DATABASES;

– створення нової бази даних: CREATE DATABASE,

```
CREATE DATABASE Ім'я_бази_даних [Параметри];
```

– вибір бази даних для використання: USE <database_name>;

– видалення бази даних: DROP DATABASE <database_name>;

– перегляд таблиць, доступних в базах даних: SHOW TABLES;

– створення нової таблиці: CREATE TABLES

```
CREATE TABLE ім'я_таблиці
    (ім'я_стовпця тип_даних [ PRIMARY KEY] [NULL | NOT NULL ] [,...n])
```

Перелік і можливі значення параметрів залежать від конкретної СКБД і найчастіше описують фізичні параметри БД, засоби управління безпекою, мовні параметри тощо.

Приклад 1. Створення БД ***Sales*** в Microsoft SQL Server:

```
USE master;
GO
CREATE DATABASE Sales
ON
( NAME = Sales_dat,
    FILENAME = 'C:\Program Files\Microsoft SQL
Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\saledat.mdf',
    SIZE = 10,
    MAXSIZE = 50,
    FILEGROWTH = 5 )
LOG ON
( NAME = Sales_log,
    FILENAME = 'C:\Program Files\Microsoft SQL
Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\salelog.ldf',
    SIZE = 5MB,
    MAXSIZE = 25MB,
    FILEGROWTH = 5MB ) ;
GO
```

*Приклад 2. Код с*творення нової таблиці ***users***:

```
CREATE TABLE users (
    id INT PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

У цьому прикладі використано DDL-операцію CREATE TABLE, щоб створити нову таблицю з назвою "users". Поля "id", "username" та "email" описані з їх типами даних та

обмеженнями, такими як NOT NULL та UNIQUE. Крім того, вказано, що значення для поля "created_at" буде за замовчуванням поточним (час на момент створення).

Далі наведено ще декілька прикладів використання команд DDL, пов'язаних з таблицями.

Приклад 3.

```
-- Створення таблиці
CREATE TABLE Users (
    id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE
);
```

Приклад 4.

```
-- Зміна структури таблиці
ALTER TABLE Users ADD COLUMN created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP;
```

Приклад 5.

```
-- Видалення таблиці
DROP TABLE Users;
```

4.6 Рішення типового завдання

Модель реляційної БД **Інститут** має вигляд:

Uchni (Код, Прізвище, Ім'я, По батькові, Дата народження, Фото, Характеристика);

Predm (Код, Назва, Кількість годин, Кількість лабораторних);

Usp (№ п/п, Код учня, Код предмету, Оцінка)

Таблиця **Usp** зв'язана з таблицями **Uchni** та **Predm** зовнішніми ключами Код учня, Код предмету.

Завдання1: створити БД *Інститут*; скласти в БД *Інститут* запит DDL для створення таблиці ***Drus* (Друзі)**.

Рішення:

CREATE TABLE drus

([kd] integer,

[pr_dr] text,

[im_dr] text,

[d_nar] date,

[tel] text,

CONSTRAINT [in1] PRIMARY KEY ([kd]));

Завдання 2: створити в базі даних **Інститут** DDL-запит, що додає до таблиці ***Drus*** (Друзі) поле ***Adr*** (Адреса).

Рішення:

ALTER TABLE drus

ADD COLUMN [adr] text;

4.7 Завдання до самостійної роботи

1. До спроектованої раніше БД, сформулювати запити, аналогічні наведеним у заданнях п. 4.6.

2. Скласти звіт, що містить формулювання і результати роботи запитів.

4.8 Контрольні питання

1. Що можна робити за допомогою SQL?

2. Чи вірно твердження, що мова SQL - непроцедурна мова?

3. Чим DDL відрізняється від DML? Як вони подані в SQL?

4. Яка команда дозволяє створити таблицю? Наведіть приклад такого запиту для таблиці з двома атрибутами

5. Які ви знаєте команди мови DDL?

6. Чому DDL команди не є транзакціями і не потребують COMMIT (примусового внесення даних в БД)?

4.9 Рекомендована література та інші джерела інформації

1. What is ODBC used for: a complete and practical guide [Електронний ресурс]. – Режим доступу: https://web.posibnyky.vntu.edu.ua/fitki/11petuh_bazdanyh_movy_zalitiv/13.htm

2. Костенко О. Б. Організація баз даних та знань : конспект лекцій (для студентів денної та заочної форм навчання першого (бакалаврського) рівня вищої освіти за спеціальністю 126 – Інформаційні системи та технології) / О. Б. Костенко, І. О. Гавриленко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2021. – 92 с. [Електронний ресурс]. – Режим доступу: http://eprints.kname.edu.ua/60505/1/2020%20%D0%BF%D0%B5%D1%87.%20134 _%D0%9B.pdf

3. Бази даних. Мови запитів, управління транзакціями, розподілена обробка даних. Базові оператори мови SQL та особливості їх запису. [Електронний ресурс]. – Режим доступу: https://web.posibnyky.vntu.edu.ua/fitki/11petuh_bazdanyh_movy_zalitiv/13.htm

4. SQL – запити. Коротка характеристика мови SQL [Електронний ресурс]. – Режим доступу: https://jetiq.vntu.edu.ua/fdb/750/SQL_%d0%97%d0%90%d0%9f%d0%98%d0%a2%d0%98.pdf

5. Вікіпедія – Вільна енциклопедія – *https://uk.wikipedia.org/wiki*

5 ПРАКТИЧНЕ ЗАНЯТТЯ № 5

5.1 Тема заняття

Мова маніпулювання даними DML. Інструкція SELECT

5.2 Мета заняття

Закріплення теоретичних знань зі створення SQL-запитів для вибору даних з БД.

5.3 Задачі заняття

Вивчення правил формування SQL-запитів для вибору даних з БД: простих, складних та вкладених SQL-запитів з використанням операторів в інструкції SELECT.

5.4 Постановка завдання

Вивчити параметри інструкції SELECT, опрацювати наведені приклади формування SQL-запитів для вибору даних з БД: прості запити, складні, підсумкові, запити з агрегованими даними і вкладеними запитами.

Для виконання практичного заняття необхідно користуватися матеріалами лекцій і літературними джерелами, наведеними у списку літератури.

5.5 Основні відомості з теорії

5.5.1 Оператор SELECT є найпоширенішим SQL-запитом, що належить до групи DML, і дозволяє вибирати дані з однієї або декількох таблиць або уявлень БД.

Оператор SELECT реалізує всі операції реляційної алгебри. У найзагальнішому вигляді оператор може бути представлений в наступному вигляді:

SELECT <що виводиться>

FROM <звідки>

WHERE <яким умовам має відповідати>

GROUP BY <колонки, за якими виконується групування даних>

HAVING <умова відбору груп>

ORDER BY <в якому порядку виводити дані>,

де SELECT– ключове слово, яке повідомляє БД, що ця команда є запитом, тобто всі запити починаються цим словом;

FROM – ключове слово, подібно до SELECT, яке повинне бути представлене в кожному запиті. Воно супроводжується пропуском і потім ім'ям таблиці яка використовується як джерело інформації.

5.5.2 Прості SQL-запити виконують одну операцію вибірки без додаткових обчислень чи вкладених запитів. Зазвичай використовують одну таблицю та прості умови.

Приклад:

```
-- Вибрати всі товари з ціною більше 100
SELECT name, price
FROM products
WHERE price > 100;
```

5.5.3 Складні SQL-запити містять кілька умов, об’єднання таблиць (JOIN), групування (GROUP BY), сортування (ORDER BY) або агрегатні функції.

Приклад:

```
-- Порахувати кількість замовлень для кожного клієнта
SELECT c.customer_id, c.name, COUNT(o.order_id) AS total_orders
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
WHERE o.status = 'completed'
GROUP BY c.customer_id, c.name
ORDER BY total_orders DESC;
```

5.5.4 Вкладені (підзапити, subqueries). Це запити, які виконуються всередині іншого запиту та використовуються для фільтрації, обчислень та передачі проміжних результатів для порівнянь у головному запиті:

–перевірки наявності результатів підзапиту (використовуються зарезервовані слова EXISTS або NOT EXISTS);

–пошуку значень в основному запиті, які дорівнюють, перевищують або менше значень, що повертаються підзапитом (використовуються зарезервовані слова ANY, IN або ALL).

Підзапит міститься усередині пропозиції WHERE іншого запиту. При цьому змінюється порядок обчислення результату запиту: спочатку обчислюється результат внутрішнього запиту (підзапиту), а потім дані, отримані в підзапиті, використовують для оцінки істинності умови зовнішнього запиту.

У підзапитах допускається використання агрегатних функцій (AVG(), COUNT(), MAX(), MIN()).

Приклад:

```
-- Знайти товари, ціна яких вища за середню ціну всіх товарів
SELECT name, price
FROM products
WHERE price > (
    SELECT AVG(price)
    FROM products
);
```

Можна використовувати підзапити для вибору рядків, які містять певне значення, якщо використовується спеціальний оператор ***IN*** (***NOT IN***) (оператори BETWEEN, LIKE, та IS NULL не можуть використовуватися з підзапитами). У будь-якій ситуації, де застосовується реляционный оператор рівності (=), можна використовувати IN.

Приклад:

Є дві таблиці – *Customers* (Клієнти) та *Orders* (Замовлення) (рис. 5.1). З таблиці *Customers* потрібна інформація про клієнтів, які розмістили замовлення. Ми вибираємо ідентифікатор клієнта (*customer_id*) та його ім'я (*first_name*) з таблиці *Customers*, лише якщо той самий ідентифікатор також є в таблиці *Orders*.

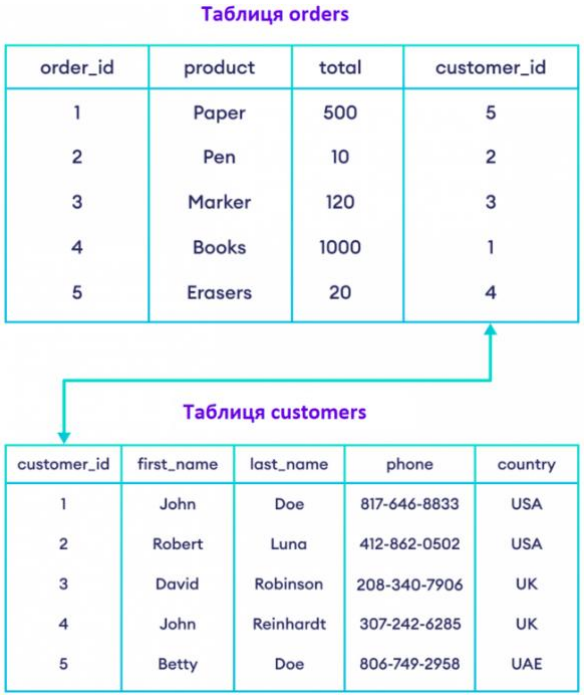


Рисунок 5.1 – Схема зв'зку між таблицями *Customers* та *Orders*

Формулювання підзапиту:

SELECT customer_id, first_name

FROM Customers

```
WHERE customer_id IN (
  SELECT customer_id
  FROM Orders
);
```

Команда ***SELECT** ** не може використовуватися у підзапиті.

Можна також використовувати вкладені запити усередині пропозиції HAVING.

Таким чином, застосування вкладених запитів з метою використання їхнього результату для керування іншим запитом розширює можливості SQL, дозволяючи виконати більша кількість функцій.

5.6 Рішення типових завданнь

Розглянемо приклади запитів на вибірку до реляційної БД (рис. 5.2), модель якої має вигляд:

АВТО (номер, марка, колір, ціна, рік_випуску, ном_посвідч_водія, ТО)

ВОДІЙ (ном_посвідч_водія, ФІО, дата_народж, адреса)

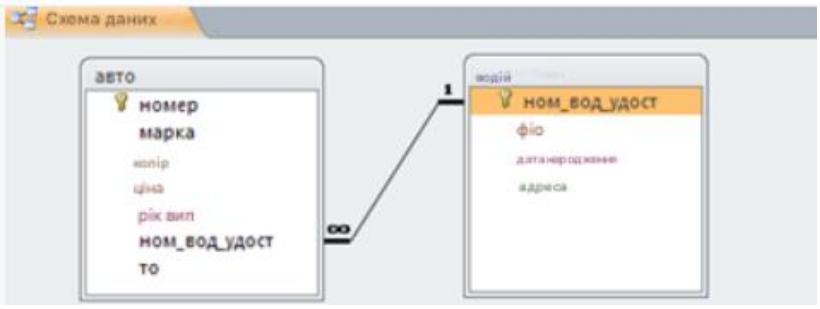


Рисунок 5.2 – Схема БД

5.6.1 Формування простих запитів на вибірку

1. Вибір авто, ціна яких лежить у зазначеному диапозону цін:

   SELECT * FROM авто WHERE ціна BETWEEN 46700 AND 85786

2. Вибір авто двох заданих марок:

   SELECT * FROM авто WHERE марка = 'lada' OR марка = 'bmw'

3. Вибір авто червоного і жовтого кольору:

   SELECT * FROM авто WHERE колір IN ('червоний', 'жовтий')

4. Вибір водіїв з прізвищем, починаються на літеру 'О':

   SELECT * FROM водій WHERE фио LIKE 'О%'

5. Підрахувати загальну КОЛ_ВО АВТО В ТАБЛИЦІ

SELECT COUNT (*) AS [КОЛ_ВО_АВТО] FROM авто

6. Підрахувати середню вартість АВТО МАРКИ 'lada':

SELECT марка, AVG (ціна) AS [СРЕДНЯЯ_ЦЕНА] FROM авто WHERE марка LIKE 'lada'

7. Вибір авто, які не пройшли тех.осмотр, і вартість яких лежить в заданому діапазоні:

SELECT * FROM авто WHERE TO = 0 AND ціна BETWEEN 46700 AND 85786

8. Вибір марок авто без повторень:

SELECT DISTINCT марка FROM авто

9. Вибір авто, мають ціну більше середньої:

SELECT * FROM авто WHERE ціна> (SELECT AVG (ціна) FROM авто)

10. Підрахувати вартість найдорожчого авто:

SELECT MAX (ціна) AS [MAX_ЦЕНА] FROM авто

11. Вибір найдорожчих авто:

SELECT * FROM авто WHERE ціна = (SELECT MAX (ціна) FROM авто)

12. Вивести інформацію, впорядкування за спаданням вартості:

SELECT * FROM авто ORDER BY ціна DESC

13. Вивести інформацію, відсортований за марками та році:

SELECT * FROM авто ORDER BY марка DESC, год_вип ASC

5.6.2 Формування складних та вкладених запитів на вибірку даних

Нижче наведені постановка завдання і його реалізація за допомогою SQL.

Завдання 1: підрахувати середню вартість авто кожній марки.

Рішення:

SELECT марка, AVG (ціна) AS [СРЕДНЯЯ_ЦЕНА] FROM авто

GROUP BY марка;

Завдання 2: підрахувати кількість авто кожній марки.

Рішення:

SELECT марка, COUNT (марка) AS [КОЛ_ВО] FROM авто

GROUP BY марка;

Завдання 3: підрахувати середній і сумарна вартість авто кожного кольори.

Рішення:

SELECT колір, AVG (ціна) AS [СРЕД_ЦЕНА],

SUM (ціна) AS [СУМ_ЦЕНА] FROM авто GROUP BY колір;

Завдання 4: вибір найдорожчого авто кожній марки.

Рішення:

SELECT марка, MAX (ціна) AS [MAX_ЦЕНА] FROM авто

GROUP BY марка;

Завдання 5: вибір власників, мають більше двох авто.

Рішення:

SELECT ПІБ, COUNT (*) AS [КОЛ_ВО_АВТО] FROM авто

GROUP BY фио HAVING COUNT (*)> 2

Завдання 6: знайти найбільшу вартість серед авто.

Рішення:

SELECT MAX(цена) FROM авто;

Завдання 7: підрахувати кількість машин червоного кольору (обчислення кількості записів, що задовольняють умові).

Рішення:

SELECT COUNT (авто.номер) FROM авто WHERE авто.цвет = “красный”;

Завдання 8: підрахувати середню вартість машин кожного року випуску.

Рішення:

SELECT AVG (авто. цена) FROM авто GROUP BY авто.год_вып;

Завдання 9: підрахувати кількість авто кожної марки, вивести ті з них, кількість яких більше 100 (угруповання з відбором груп за умовою).

Рішення:

SELECT авто.марка, COUNT(авто.номер) FROM авто

GROUP BY авто. марка HAVING COUNT (авто.номер) > 100;

Завдання10: вибрати всіх власників авто, чиї автомобілі пройшли техогляд.

Рішення:

SELECT * FROM водитель WHERE водитель.ном_вод_удост IN

(SELECT авто.ном_вод_удост FROM авто WHERE авто.то =1).

Завдання11: вибрати авто, що мають ціну менше середньої й випущених після 2000 року.

Рішення:

SELECT * FROM авто

WHERE цена <(SELECT AVG(цена) FROM авто)

AND год_вып>#01/01/2001#;

Завдання12: якщо в базі даних не існує інформації про автомобілі, випущених раніше 1970 року, то видати всю інформацію про авто, що зареєстрований у базі даних раніше інших, і про його власника.

Рішення:

```
SELECT TOP 1 авто.*, водитель.фио, водитель.адрес
FROM авто, водитель
WHERE NOT EXISTS (SELECT * FROM авто
WHERE авто.год_вып <=1970)
AND авто.ном_вод_удост = водитель.ном_вод_удост
  ORDER BY авто.год_вып ASC;
```

Завдання13: показати найбільшу ціну серед автомобілів і для кожного авто показати на скільки дане авто дешевше найдорожчого (додаткові запити в пропозиції SELECT).

Рішення:

```
SELECT (SELECT Max(авто.цена) FROM авто) AS [Найбільша ціна авто],
[Найбільша ціна авто] – авто.цена AS [На скількі дешевше],
авто.марка AS [Назва], авто.год_вып AS [Рік випуску]
FROM авто.
```

5.7 Завдання до самостійної роботи

1. До спроектованої на попередніх заняттях БД, сформувати прості запити на вибірку даних з кожної таблиці, вкладені запити, а також підсумкові запити за наведеними прикладами завдань п. 5.6.
2. Реалізувати у СКБД сформовані SQL-запити.
3. Скласти звіт, що містить постановку завдань на вибірку даних з таблиць БД, сформовані запити й результати їх роботи.

5.8 Контрольні питання

1. Як задаються умови відбору в SQL-Запитах?
2. Для чого використовуються умови вибірки? Які можливості надає SQL для формування умов
3. Як може використовуватися пропозиція SELECT?
4. Поясніть повну конструкцію SELECT.
5. Що відображає агреговане значення?
6. Поясніти порядок створення підсумкового запиту.
7. Поясніти порядок створення перехресного запиту.

5.9 Рекомендована література та інші джерела інформації

1. SQL – запити. Коротка характеристика мови SQL [Електронний ресурс]. – Режим доступу: https://jetiq.vntu.edu.ua/fdb/750/SQL_%d0%97%d0%90%d0%9f%d0%98%d0%a2%d0%98.pdf

2. Згуровська Л.П. Бази даних. Комп'ютерний практикум [Електронний ресурс]: https://ela.kpi.ua/items/1b7e26c2-dd9c-43fb-aaba-938867a3fd76

3. Костенко О. Б. Організація баз даних та знань : конспект лекцій (для студентів денної та заочної форм навчання першого (бакалаврського) рівня вищої освіти за спеціальністю 126 – Інформаційні системи та технології) / О. Б. Костенко, І. О. Гавриленко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2021. – 92 с. [Електронний ресурс]. – Режим доступу: http://eprints.kname.edu.ua/60505/1/2020%20%D0%BF%D0%B5%D1%87.%20134 _%D0%9B.pd

4. Список ресурсів по Microsoft SQL Server [Електронний ресурс]. – Режим доступу: https://habr.com/post/305866/.

5. Вікіпедія - Вільна енциклопедія – https://uk.wikipedia.org/wiki

6 ПРАКТИЧНЕ ЗАНЯТТЯ № 6

6.1 Тема заняття

Вибірка даних з декількох зв'язаних таблиць

6.2 Мета заняття

Закріплення теоретичних знань мови SQL і придбання навичок створення запитів даних з декількох зв'язаних таблиць.

6.3 Задачі заняття

Виконання операцій створення запитів даних з декількох зв'язаних таблиць

6.4 Постановка завдання

Створити запити даних з декількох зв'язаних таблиць, використвуючи вкладені запити або операцию INNER JOIN.

Для виконання практичного заняття необхідно користуватися матеріалами лекцій і літературними джерелами, наведеними у списку літератури.

6.5 Основні відомості з теорії

Найважливішою особливістю запитів SQL є їх здатність визначати зв'язки між декількома таблицями і виводити інформацію з них в термінах цих зв'язків. Такі операції називаються об'єднанням, яке є одним з видів операцій в реляційних БД – адже це є основою реляційного підходу до зберігання даних в таблицях. Використовуючи об'єднання, відбувається безпосереднє зв'язування інформації із будь-яким номером таблиці, і, таким чином, створюються зв'язки між порівнянними частинами даних.

При багатотабличному запиті, таблиці, представлені у вигляді списку в пропозиції FROM, відокремлюються одна від одної комами. Предикат запиту може посилатися до будь-якого стовпця будь-якої зв'язаної таблиці і, отже, може використовуватися для зв'язку між ними.

Звичайно предикат порівнює значення в стовпцях різних таблиць, щоб визначити, чи задовольняє WHERE встановленій умові. До цього імена таблиць в запитах опускалися, тому

запрошувалася тільки одна таблиця одночасно. Навіть при запиті із декількох таблиць допускається опускати їх імена, якщо, звичайно, вони різні. Тепер же виникає необхідність використання імен стовпців і таблиць, оскільки в багатотабличному запиті можуть виникати неоднозначності.

При об'єднанні таблиць, добре проглядається логіка зв'язку між даними і можна більше не обмежуватися переглядом однієї таблиці в кожен момент часу. Крім того, об'єднання дозволяє робити складні порівняння між будь-якими полями будь-якого числа таблиць і використовувати отримані результати для того, щоб вирішувати – яку інформацію хотілося б бачити.

Створити запит даних з декількох зв'язаних таблиць можно з використанням вкладених запитів. У найзагальнішому випадку, запити можуть управляти іншими запитами – це робиться шляхом розміщення запиту всередину іншого предиката, який використовує вивід внутрішнього запиту для встановлення вірного або невірного значення предиката.

Після ознайомлення з роботою підзапитів, розглянемо використання спеціальних операторів EXISTS, ANY, ALL, I SOME, які завжди беруть подзапросы як аргументи.

Оператор EXISTS використовується для вказівки предикату на те, щоб робити або не робити вивід у подзапросе, при цьому EXISTS дає як результат значення ІСТИНА або НЕПРАВДА. Він може працювати в предикаті або в комбінації з іншими булевскими вираженнями - AND, OR, і NOT. EXISTS бере подзапрос як аргумент і оцінює його як щирий, якщо він здійснює будь-який вивід, або як помилковий, якщо він не робить цього.

Оператори SOME і ANY досить часто взаємозамінні. У кожному разі запит, що може бути реалізований з ANY і ALL, може бути також сформульований з EXISTS, але не навпаки. У ряді випадків, через розходження в обробці NULL значень, варіант із EXISTS не абсолютно ідентичний, чим при використанні ANY, тому необхідно в таких випадках застосовувати команду IS NULL.

Основна перевага у використанні ANY і ALL у порівнянні з EXISTS полягає в тім, що останній вимагає співвіднесених подзапросов, часом складних для розуміння. Крім того, подзапросы з ANY або ALL можуть виконуватися один раз і мати вивід, використовуваний для визначення предиката для кожного рядка основного запиту, а EXISTS вимагає, щоб весь подзапрос повторно виконувався для кожного рядка основного запиту.

Оператор ALL працює таким чином, що предикат є вірним, якщо кожне значення, обране подзапросом, задовольняє умові в предикаті зовнішнього запиту.

Найпоширеніший тип об'єднання таблиць - це операція INNER JOIN, яку можна використовувати в будь-якій пропозиції FROM.

Операція INNER JOIN об'єднує записи із двох таблиць, якщо сполучні поля цих таблиць містять однакові значення.

Синтаксис:

FROM таблиця1 INNER JOIN таблиця2 ON таблиця1.поле1

оператор_ порівняння таблиця2.поле2

Компоненти, що становлять операцію INNER JOIN показані у таблиці 8.1.

Таблиця 8.1 – Компоненти INNER JOIN

Аргумент	Опис
таблиця1, *таблиця2*	Імена таблиць, з яких поєднуються записи
поле1, *поле2*	Імена полів, що зв'язуються. Якщо поля не містять числових даних, вони повинні ставитися до одного типу даних і містити деякі дані. Імена цих полів можуть бути різними
оператор_ порівняння	Будь-який оператор порівняння: «=», «<», «>», «<=», «>=» або «<>»

6.6 Рішення типового завдання. Приклади

Розглянемо приклади побудови богатотабличних запитів до бази даних, представленої на рисунку 8.1.

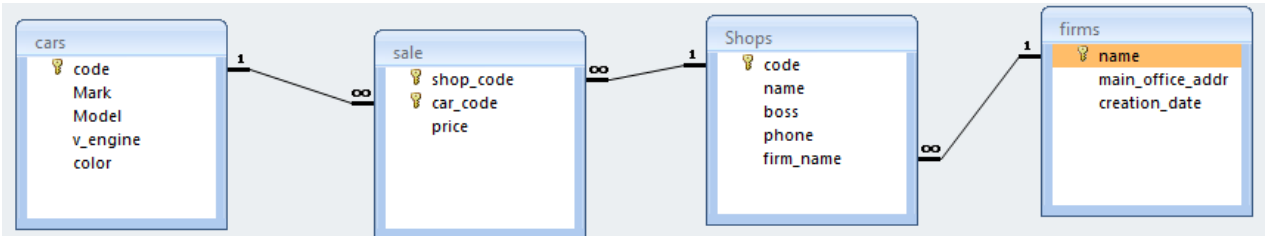


Рисунок 8.1 - Схема використовуваної БД

Нижче наведені постановка завдання і його реалізація за допомогою SQL.

1. Запит на створення таблиці (SELECT INTO):

SELECT * INTO new_cars FROM cars

2. Запит з об'єднанням

SELECT mark, model FROM cars

UNION SELECT name, boss FROM shops

3. Запит з с виключенням повторюваних рядків:

SELECT DISTINCT mark, model FROM cars

4. Прості богатотабличні запити

4.1. Вибір всіх магазинів, чиї фірми власники відкрилися після 01.01.2008:

SELECT * FROM shops WHERE firm_name IN (SELECT name FROM firms WHERE creation_date < #01/01/2008#)

4.2. Вибір магазинів і машин, проданих за ціною дорожче 90000, із сортуванням записів за назвою магазина:

SELECT shops.[name], cars.* FROM shops, cars WHERE shops.code IN (SELECT shop_code FROM (SELECT shop_code,car_code FROM sale WHERE price > 90000)) AND cars.code IN (SELECT car_code FROM sale WHERE sale.shop_code = shops.code AND price>90000) ORDER BY shops.name

4.3. Вибір продаваних машин:

SELECT DISTINCT cars.* FROM cars Inner Join sale ON sale.car_code = cars.code

SELECT cars.* FROM cars WHERE EXISTS(SELECT * FROM sale WHERE sale.car_code = cars.code)

SELECT cars.* FROM cars WHERE code NOT IN (SELECT car_code FROM sale)

4.4. Вибір магазинів і машин, продаваних за ціною нижче 100000:

SELECT shops.name, cars.*, sale.price FROM (cars INNER JOIN sale ON sale.car_code = cars.code) INNER JOIN shops ON shops.code= sale.shop_code WHERE sale.price <100000

4.5. Вибір машин із вказівкою ціни продажу в кожному магазині, якщо машина ніде не продається - тільки інформації про неї самої:

SELECT shops.name, cars.*, sale.price FROM (sale INNER JOIN shops ON shops.code= sale.shop_code) RIGHT JOIN cars ON sale.car_code = cars.code

4.6. Вибір машин із вказівкою середньої вартості продажів:

SELECT cars.mark,cars.model, cars.v_engine,cars.color, AVG(sale.price) FROM cars LEFT JOIN sale ON cars.code = sale.car_code GROUP BY cars.mark, cars.model, cars.v_engine, cars.color

SELECT cars.mark,cars.model, cars.v_engine,cars.color, (select AVG (sale.price) from sale Where sale.car_code = cars.code) FROM cars

6.7 Завдання до самостійної роботи

1. Вибрати всі машини з об’ємом двигуна > 1,4 у нову таблицю, упорядкувавши по марці й моделі.
2. Вибрати в нову таблицю всі машини червоного, чорного й жовтого коляру.
3. Вибрати всі магазини, які хоч щось продають (можна використати Inner Join+ Distinct, Exists, IN).

6.8 Контрольні питання

1. Які операції називаються об'єднанням?
2. Як представлені таблиці при багатотабличному запиті?
3. У чому особливості багатотабличних запитів?
4. Визначити типи об'єднання таблиць.
5. Використання вкладених запитів для об'єднання таблиць.
6. Для чого використовуються операторі EXISTS, ANY, ALL і SOME?

6.9 Рекомендована література та інші джерела інформації

1. Згуровська Л.П. Бази даних. Комп'ютерний практикум [Електронний ресурс]: https://ela.kpi.ua/items/1b7e26c2-dd9c-43fb-aaba-938867a3fd76

2. Костенко О. Б. Організація баз даних та знань : конспект лекцій (для студентів денної та заочної форм навчання першого (бакалаврського) рівня вищої освіти за спеціальністю 126 – Інформаційні системи та технології) / О. Б. Костенко, І. О. Гавриленко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2021. – 92 с. [Електронний ресурс]. – Режим доступу: http://eprints.kname.edu.ua/60505/1/2020%20%D0%BF%D0%B5%D1%87.%20134 _%D0%9B.pd

3. Список ресурсів по Microsoft SQL Server [Електронний ресурс]. – Режим доступу: https://habr.com/post/305866/.

4. Вікіпедія - Вільна енциклопедія – https://uk.wikipedia.org/wiki

7 ПРАКТИЧНЕ ЗАНЯТТЯ № 7

7.1 Тема заняття

Команди модифікації даних (DELETE, INSERT, UPDATE)

7.2 Мета заняття

Закріплення теоретичних знань мови SQL і придбання навичок видалення, вставки й відновлення даних

7.3 Задачі заняття

Виконання операцій модифікації даних

7.4 Постановка завдання

Побудувати SQL запит вставки, запит на видалення записів за критерієм і запит на модифікацію значень полів записів за заданим критерієм.

Для виконання практичного заняття необхідно користуватися матеріалами лекцій і літературними джерелами, наведеними у списку літератури.

7.5 Основні відомості з теорії

При роботі з SQL виключно важливо не тільки уміти вибирати дані, але і користуватися засобами, які управляють значеннями в таблиці. Значення можуть бути поміщені і видалені з полів трьома командами мови DML (Мова Маніпулювання Даними), а саме:

- INSERT – вставити;
- UPDATE – модифікувати;
- DELETE – видалити.

Команда **DELETE** може видаляти тільки цілі записи таблиці, а не індивідуальні значення того або іншого поля. З цієї причини для даного оператора параметр поля є недоступним.

Команда DELETE має формат:

DELETE

FROM {базова таблиця | подання}

[WHERE фраза];

і дозволяє видалити вміст всіх рядків зазначеної таблиці (при відсутності WHERE фрази) або тих її рядків, які виділяються WHERE фразою.

Команда INSERT не здійснює жодного виводу, але бажано, щоб СКБД давала деяке підтвердження того, що дані були успішно внесені. Крім того, слід пам'ятати, що ім'я таблиці, в яку здійснюється вставка, повинно бути заздалегідь означене, а кожне значення в списку даних, що вставляються, повинне співпадати з типом даних стовпця, в який воно вставляється.

Значення в цьому списку вводяться в таблицю в тому порядку, в якому вони записані в команді, тому перше значення автоматично потрапляє в перший стовпець, друге – в другий стовпець і т.д.

Якщо потрібно ввести в таблицю NULL значення, то воно вводиться точно так, як і звичайне.

Команда **INSERT** має один з наступних форматів:

INSERT

INTO {базова таблиця | подання} [(поле [,поле] ...)]

VALUES ({константа | змінна} [,{константа | змінна}] ...);

або

INSERT

INTO {базова таблиця | подання} [(поле [,поле] ...)]

підзапит;

У першому форматі в таблицю вставляється рядок зі значеннями полів, зазначеними в переліку фрази VALUES (значення), причому і-е значення відповідає і-му стовпцю в списку стовпців (стовпці, не зазначені в списку, заповнюються NULL-Значеннями). Якщо в списку VALUES фрази зазначені всі стовпці модифікується таблицы, що, і порядок їхнього перерахування відповідає порядку стовпців в описі таблиці, то список стовпців у фразі INTO можна опустити.

Набагато серйознішим питанням є можливість зміни деяких або всіх значень в існуючому рядку таблиці, що реалізується за допомогою команди **UPDATE**.

Ця команда містить ключове слово UPDATE, де вказується ім'я використовуваної таблиці, і пропозиція SET, яка визначає зміну, що вноситься, для необхідного поля таблиці.

Наприклад, щоб змінити оцінки всіх студентів на 5, необхідно використати команду:

UPDATE USP

SET OCINKA = 5;

Звичайно, набагато частіше доводиться вказувати не все, а тільки певні рядки таблиці для зміни єдиного значення, і з цією метою разом з UPDATE можна використовувати предикати.

Якщо у фразі WHERE пропозицій DELETE і UPDATE використовується вкладений подзапрос, то у фразі FROM цього подзапроса не повинна згадуватися таблиця, з якої віддаляються (у якій обновляються) рядки. Аналогічно, у подзапросе пропозиції INSERT не повинна згадуватися таблиця, у яку завантажуються дані.

7.6 Рішення типового завдання

Всі наступні приклади наведені для таблиць, які показані на рисунку 9.1.

АВТО (**Номер**, Марка, Колір, Рік_випуску, Ціна, V_двигуна, Дата_ТО, Відділення_МВС)

МВС (**Відділення**, Область, Адреса, Начальник)

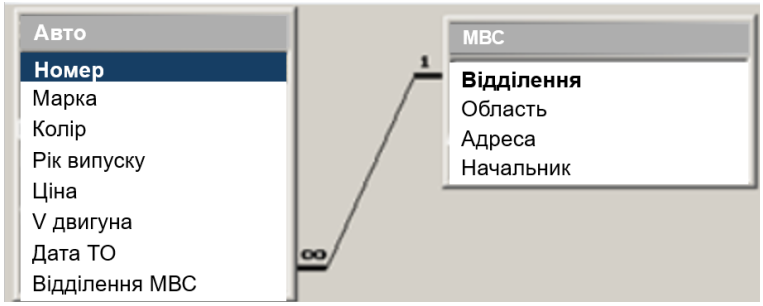


Рисунок 9.1 – Схема використовуваної БД

Нижче наведені постановка звдання і його реалізація за допомогою SQL.

1. Видалення всіх записів з таблиці **Авто**:

```
DELETE FROM  Авто;
```

2. Видалення конкретного авто (єдиного запису):

```
DELETE FROM  Авто WHERE [Номер] LIKE "BB1234AA";
```

3. Видалення всіх червоних авто раніше 2000 року випуску (видалення групи записів):

```
DELETE FROM  Авто WHERE [Колір] LIKE "червоний" AND [Рік_випуску] < 2000;
```

4. Видалення всіх авто, зареєстрованих у Житомирській та Черкаській областях (з вкладеним підзапитом):

```
DELETE FROM  Авто
    WHERE [Відділення_МВС] IN (SELECT MBC.[Відділення] FROM MBC
    WHERE MBC.[Область] LIKE "Черкаська" OR  MBC.[Область] LIKE "Житомирська");
```

або

```
DELETE Авто.*
    FROM MBC INNER JOIN Авто ON MBC.[Відділення]=Авто.[Відділення_MBC]
    WHERE MBC.[Область] LIKE "Черкаська" OR  MBCO.[Область] LIKE "Житомирська";
```

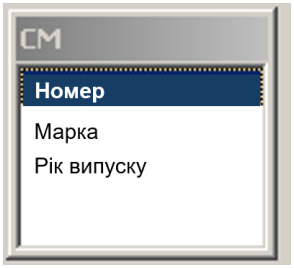
5. Вставка запису в таблицю Авто:

```
INSERT INTO Авто ([Номер],[Марка],[Колір],[Відділення_MBC])
    VALUES ("BB5678AH","ВАЗ-21099","жовтий",1);
```

6. Вставка єдиного запису без переліку списку стовпців:

```
INSERT INTO Авто
    VALUES ("BB9854AH","ВАЗ-2107","зелений",2015,15000,1.3, #04/06/2026#,1);
```

7. Вставка декілька записів у таблицю СМ (Службові машини):




```
INSERT INTO СМ
SELECT Авто.[Номер],Авто.[Марка],Авто.[Рік_випуску]
FROM Авто WHERE Авто.[Номер] LIKE “*AH”;
```

8. Оновлення всіх записів:

```
UPDATE Авто SET [Цвет] = "чорний";
```

9. Збільшити на 20% вартість машин, у яких вона мінімальна:

```
UPDATE Авто SET [Ціна] = [Ціна]*1.2
    WHERE [Ціна] = (SELECT MIN([Ціна]) FROM Авто);
```

10. Встановити дату проходження ТО всім авто, зареєстрованим у Львівській області, 2015 року випуску:

```
UPDATE Авто SET [Дата_ТО] = #04/01/2026#
    WHERE [№ отд МРЭО] IN (SELECT МРЭО.[№ отделения]
    FROM МРЭО WHERE МРЭО.[Область] LIKE "Луганская")
    AND Машины.[Год выпуска] = 2015;
```

7.7 Завдання на самостійну роботу

До спроектованої на попередніх заняттях бази даних, побудувати SQL запит вставки, запит на видалення записів за критерієм і запит на модифікацію значень полів записів за заданим критерієм.

Скласти звіт, що містить формулювання й результати роботи запитів.

7.8 Контрольні питання

1. Які команди використовуються для модифікації даних?
2. Поясніть призначення і формат команди INSERT.
3. Поясніть призначення іформат команди UPDATE.
4. Поясніть призначення і формат команди DELETE.

7.9 Рекомендована література та інші джерела інформації

1. Згуровська Л.П. Бази даних. Комп'ютерний практикум [Електронний ресурс]: https://ela.kpi.ua/items/1b7e26c2-dd9c-43fb-aaba-938867a3fd76

2. Костенко О. Б. Організація баз даних та знань : конспект лекцій (для студентів денної та заочної форм навчання першого (бакалаврського) рівня вищої освіти за спеціальністю 126 – Інформаційні системи та технології) / О. Б. Костенко, І. О. Гавриленко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2021. – 92 с. [Електронний ресурс]. – Режим доступу: http://eprints.kname.edu.ua/60505/1/2020%20%D0%BF%D0%B5%D1%87.%20134 _%D0%9B.pd

3. Список ресурсів по Microsoft SQL Server [Електронний ресурс]. – Режим доступу: https://habr.com/post/305866/.

4. Вікіпедія - Вільна енциклопедія – https://uk.wikipedia.org/wiki

8 ПРАКТИЧНЕ ЗАНЯТТЯ № 8

8.1 Тема заняття

Збережені процедури

8.2 Мета заняття

Закріплення теоретичних знань по створенню й використанню збережених процедур (CREATE PROCEDURE, ALTER PROCEDURE , DROP PROCEDURE0

8.3 Задачі заняття

Виконання операцій по створенню й використанню збережених процедур

8.4 Постановка завдання

Створити й використувати збережені процедури у середовищи SQL-Server.

Для виконання практичного заняття необхідно користуватися матеріалами лекцій і літературними джерелами, наведеними у списку літератури.

8.5 Основні відомості з теорії

Збережені процедури (SP - Stored Procedures) - це підпрограми мовою T-SQL, що зберігаються в БД, до яких можна звертатися із зовнішніх додатків, запитів і інших збережених процедур. Збережена процедура може приймати дані через параметри, виконувати інструкції мовою T-SQL і повертати результат у вигляді результуючих наборів даних і/або через вихідні (output) параметри.

Параметри процедури описуються в такий спосіб:

{ім'я параметра (починається з @)} {тип параметра (такі ж, як і типи стовпців)} [output (якщо параметр вихідної)]. Кілька параметрів розділяються між собою комами.

Тіло процедури складається з операторів T-SQL:

- оператори запитів (SELECT/INSERT/UPDATE/DELETE);
- оператори оголошення локальної змінної

DECLARE { ім'я змінної (починається з @)} {тип змінної };

- оператор присвоювання

SET { ім'я змінної } = {вираз типу, відповідного типу змінної }

Вираз може містити інші змінні, арифметичні операції, константи, стандартні функції, запити, які повертають єдине значення (записуються в круглих дужках) і т.і.

SELECT { ім'я змінної 1} = { ім'я стовпця 1}, { ім'я змінної 2} = { ім'я стовпця 2} и т.і. FROM и т.і. – записує в змінні результат виконання запиту, що повертає одну результуючу рядок. Замість імені стовпця може використовуватися вираз (як при створенні обчислюваних стовпців) або агрегатна функція (при підсумкових запитах і запитах з угрупованням).

– умовний оператор

IF { логічне вираження } THEN {оператор} [ELSE {оператор}]

Оператор може бути простим або складним. Складний укладається між словами BEGIN і END.

Логічний вираз може складатися з інших виразів, об'єднаних операціями порівняння (>, <, =, <>,> =, <=), логічними операціями (AND, OR, NOT), операція EXISTS ({запит на вибірку}), операція перевірки вираження на null ({вираз} is null).

– оператор циклу

WHILE {логічне вираз} {оператор}

– оператор виходу з процедури - RETURN і переривання циклу - BREAK.

– оператор виклику процедури EXEC {ім'я процедури} [параметр 1, параметр 2 ...].–

Стандартні функції (неповний перелік):

– математичні функції (напр, sin, cos, sqrt і т.д.);

– функції роботи з рядками:

1) LEN (рядковий вираз) - довжина рядка;

2) SUBSTRING (строкове вираження, початкова позиція, кол-во символів) - виділення підрядка (символи в рядках нумеруються з 1);

3) REPLACE (рядковий вираз 1, рядковий вираз 2, рядковий вираз 3) - замінює все входження підрядка, заданої рядковим виразом 2, в рядку, заданої вираженням 1, на підрядок, задану виразом 3. Функція повертає результат заміни.

Решта функцій дивитися в довідковій документації по T-SQL, що поставляється з SQL Server'ом.

– функції роботи з датою-часом:

1) GETDATE () - повертає поточну системну дату і час у форматі значень datetime;

2) DATEADD (розділ дати, число, дата) - повертає нове значення datetime, додаючи до зазначеної дати проміжок часу. Разделы даты (year – год; month – месяц; day – день; week – неделя; hour – час; minute – минута; second – секунда; millisecond – миллисекунда).

Наприклад, DATEADD(month, 7, '17.08.2006'), поверне дату 17.03.2007;

DATEPART (розділ дати, дата) – повертає вказану частину дати;

DATEDIFF (розділ дати, початкова дата, кінцева дата) - повертає кількість границь дати і часу, що перетинаються між двома зазначеними датами, іншими словами різниця між датами в одиницях розділу дати.

- функція перетворення типів

CONVERT (тип даних, вираз) - перетворює вираз в заданий тип даних.

Оператор виклику процедури

EXEC {ім'я процедури} [параметр 1, параметр 2 …]

8.6 Рішення типового завдання

```
-- ============================================
Вибірка студентів, з сортуванням по групі та імені--
============================================
CREATE PROCEDURE GetStudentsOrdered
AS
    SELECT * FROM Students ORDER BY [Group], [Name];

-- ============================================
Додавання студента-- ============================================
CREATE PROCEDURE AddStudent
    @name nvarchar(64),
    @group nvarchar(64),
    @mark int
AS
    INSERT INTO Students ([Name], [Group], Mark) VALUES (@name, @group, @mark)
-- ============================================
Видалення групи-- ============================================
CREATE PROCEDURE RemoveGroup
    @group nvarchar(64)
AS
    DELETE FROM Groups WHERE [Group] = @group
-- ============================================
Зміна атрибутів групи-- ============================================
CREATE PROCEDURE ModifyGroup
```

```
    @old_group nvarchar(64),
    @group nvarchar(64),
    @header nvarchar(50)
AS
    UPDATE Groups SET [Group] = @group, Header = @header WHERE [Group] = @old_group

    -- ============================================
    Отримання куратора і кол-ва студентів групи--
============================================
    CREATE PROCEDURE GetGroupAttributes
    @group nvarchar(64),
    @header nvarchar(50) output,
    @studentsCount int output
    AS
    BEGIN
    SELECT @header = Header FROM Groups WHERE [Group] = @group;
    SELECT @studentsCount = count(*) FROM Students WHERE [Group] = @group;
    END
```

Для получения результата работы процедуры необходимо описать выходные переменные, вызвать процедуру и вывести результаты.

```
    GO
    declare @header1 nvarchar(50);
    declare @studentsCount1 int;
    exec GetGroupAttributes 'эа',@header1 output,@studentsCount1 output;
    print 'КУРАТОР '+@header1;
    print 'КІЛЬКІСТЬ СТУДЕНТІВ В ГРУПІ '+str(@studentsCount1);
    -- ============================================
    Розформування групи @sourceGroup
    -- (навпіл в @targetGroup1 и @targetGroup2)
    -- ============================================
    CREATE PROCEDURE SeparateGroup
    @sourceGroup nvarchar(64),
    @targetGroup1 nvarchar(64),
    @targetGroup2 nvarchar(64)
```

```
AS
BEGIN
    -- Вибірка кількості студентів в видаляється групі
    DECLARE @studentsCount int;
    SELECT @studentsCount = count(*) FROM Students WHERE [Group] = @sourceGroup;
    if @studentsCount <> 0
    BEGIN
        -- отримання половини кількості студентів
        DECLARE @halfCount int;
        SET @halfCount = @studentsCount/2;
        -- Зміна групи для першої половини студентів
        UPDATE TOP (@halfCount) FROM Students SET [Group] = @targetGroup1
                WHERE [Group] = @sourceGroup ;
        -- Зміна групи для решти студентів
        UPDATE Students
                SET [Group] = @targetGroup2 WHERE [Group] = @sourceGroup;
        -- видалення вихідної групи
        EXEC RemoveGroup @sourceGroup;
    END
END
-- ============================================
Створення нової групи, в яку переводяться по третині
найгірших студентів з двох вихідних груп--
============================================
CREATE PROCEDURE NewGroup
    @newGroupName nvarchar(64),
    @newGroupHeader nvarchar(64),
    @sourceGroupName1 nvarchar(64),
    @sourceGroupName2 nvarchar(64)
AS
BEGIN
    insert into Groups ([group], header) values (@newGroupName, @newGroupHeader);
    update Students set [group] = @newGroupName where id in (select TOP(33.3) PERCENT
id from students where [group]= @sourceGroupName1 order by mark
        union
```

```
    select TOP(33.3) PERCENT id from students where [group]= @sourceGroupName2 order
by mark)
    END
```

8.7 Завдання на самостійну роботу

1. Написати збережені процедури, що виконують читання, відновлення, додавання й видалення данных кожної з таблиць. Процедура додавання повинна містити параметри для кожного стовпця таблиці. Процедура відновлення повинна містити параметри для нових значень всіх стовпців і старих значень ключових стовпців. Процедура видалення повинна містити параметри для значень ключових стовпців.
2. Написати збережену процедуру, що повертає один запис із будь-якої таблиці по ключі, переданому через параметри, через вихідні параметри.
3. Написати процедуру розформування групи. Процедуру створення нової групи, куди переводяться студенти із двох зазначених груп так, щоб у новій групі виявилася третина (самих невстигаючих) студентів вихідних груп. Вихідні групи не можна залишати без студентів.
4. Виконати всі створені збережені процедури.

8.8 Контрольні питання

1. Що таке збережені процедури?
2. Для чого створюються збережені процедури?
3. Як викликати збережену процедуру?
4. Як описуються параметри процедури?
5. З яких операторів складається тіло процедури?

8.9 Рекомендована література та інші джерела інформації

1. М.В. Добролюбова. Програмування баз даних. Конспект лекцій. – Київ: КПІ ім. Ігоря Сікорського, 2021. – 275 с.
2. Anthony DeBarros, Practical SQL, 2nd Edition. A Beginner's Guide to Storytelling with Data. Електронний ресурс: http://projanco.com/Library/Practical%20SQL%20A%20Beginner%E2%80%99s%20Guide%20to%20Storytelling%20with%20Data.pdf
3. Вікіпедія - Вільна енциклопедія – https://uk.wikipedia.org/wiki

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ

до практичних робіт з дисципліни

«ОРГАНІЗАЦІЯ БАЗ ДАНИХ»

(для здобувачів вищої освіти усіх форм навчання за спеціальностями

"Комп’ютерні науки", "Комп’ютерна інженерія")

Укладач:

Шумова Лариса Олександрівна

Редактор: Л.О. Шумова
Техн. редактор: Л.О. Шумова
Оригінал – макет: Л.О. Шумова

Підписано до друку _____
Формат 16 60 ×84 1 . Папір типограф. Гарнитура Times.
Друк офсетний. Ум. друк. арк.___. Обл.-вид.арк._____.
Тираж ___ прим. Вид. №_____. Замовл. №_____. Ціна договірна.

Видавництво Східноукраїнського національного університету імені Володимира Даля
Свідоцтво про реєстрацію: серія ___ № _____ від __.__.____ р.
Адреса університету: вул. Іоанна Павла II, 17,
м. Київ, 01042, Україна
e-mail: vidavnictvoSNU.ua@gmail.com