**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ**

# ПРОЄКТУВАННЯ БАЗ ДАНИХ ТА БАЗ ЗНАНЬ

**Методичні рекомендації до самостійної роботи студентів**
**спеціальності 186 "Видавництво та поліграфія"**
**першого (бакалаврського) рівня**

**Харків**
**ХНЕУ ім. С. Кузнеця**
**2022**

УДК 004.65(07.034)
П79

**Укладач** А. С. Гордєєв

Затверджено на засіданні кафедри комп'ютерних систем і технологій.
Протокол № 5 від 15.11.2021 р.

*Самостійне електронне текстове мережеве видання*

**Проєктування** баз даних та баз знань [Електронний ресурс] :
П79 методичні рекомендації до самостійної роботи студентів спеціальності 186 "Видавництво та поліграфія" першого (бакалаврського) рівня / уклад. А. С. Гордєєв. – Харків : ХНЕУ ім. С. Кузнеца, 2022. – 35 с.

Подано основні положення щодо організації та виконання самостійної роботи. Уміщено загальні положення щодо виконання самостійної роботи з навчальної дисципліни та програму виконання самостійної роботи. Наведено детальний опис завдань для самостійної роботи та перелік необхідної для виконання завдань літератури.

Рекомендовано для студентів спеціальності 186 "Видавництво та поліграфія" першого (бакалаврського) рівня всіх форм навчання.

**УДК 004.65(07.034)**

# Вступ

Навчальна дисципліна "Проєктування баз даних та баз знань" належить до групи нормативних навчальних дисциплін циклу професійно-орієнтованих дисциплін та вивчається згідно з навчальним планом підготовки за спеціальністю 186 "Видавництво та поліграфія".

**Мета навчальної дисципліни**: формування у студентів фундаментальних теоретичних знань з архітектури баз даних та баз знань систем оброблення інформації стадій розроблення та експлуатації електронних мультимедійних видань, їх принципів організації та особливостей використання; здобуття навичок проєктування, упровадження та використання технологічних рішень систем, що базуються на використанні баз даних та баз знань.

**Завданням навчальної дисципліни** є вивчення теоретичних основ і технологій створення баз даних, особливостей роботи над спеціальними інформаційними джерелами та інструментальними засобами, що дозволяють більш ефективно застосовувати сучасні автоматизовані системи управління базами даних.

**Об'єктом вивчення навчальної дисципліни** є технології роботи з базами даних на платформі .NET Framework.

**Предметом вивчення навчальної дисципліни** бази даних та знань у інформаційні системи стадій розроблення та експлуатації електронних мультимедійних видань.

**Інструментальною базою вивчення навчальної дисципліни** є сучасне програмне забезпечення для створення і управління базами даних *Microsoft SQL Server Management Studio* 2018 та *Visual Studio* 2019.

Необхідність здобуття розгорнутих знань із навчальної дисципліни "Проєктування баз даних та баз знань" для успішного виконання подальшої професійної діяльності й обмеженість навчального (зокрема аудиторного) навантаження студентів спеціальності 186 "Видавництво та поліграфія" зумовлює характер позааудиторної самостійної роботи студентів у межах цієї навчальної дисципліни. Виконання завдань для позааудиторної самостійної роботи має забезпечувати набуття студентами нових компетентностей у межах тієї тематики, що розглядається на лекційних і лабораторних заняттях.

Отже, **основною метою позааудиторної самостійної роботи** з навчальної дисципліни є надання можливості закріплення компетентностей студентів спеціальності 186 "Видавництво та поліграфія".

З огляду на зазначений характер і специфіку позааудиторної самостійної роботи (виду діяльності, що формує нові знання у студента без безпосередньої участі викладача), ці методичні рекомендації містять програму самостійної роботи з навчальної дисципліни, що дозволяє студенту отримати узагальнене уявлення про зміст самостійної роботи та запланувати її виконання відповідно до методичних вимог.

Виконання завдань для самостійної роботи оцінюватиметься за такими критеріями:

- повнота врахування вимог до виконання завдання;
- логічність викладеного матеріалу;
- наявність і повнота розгляду ключових понять предметної галузі завдання;
- ілюстрування опрацьованого матеріалу власними прикладами;
- наявність і обґрунтованість підсумкових висновків студента.

# Компетентності студентів спеціальності 186 "Видавництво та поліграфія" і зміст самостійної роботи

У результаті вивчення навчальної дисципліни студент має набути таких компетентностей (табл. 1).

Таблиця 1

## Компетентності та результати навчання за навчальною дисципліною

| Компетентності | Результати навчання |
|---|---|
| 1 | 2 |
| ЗК-2. Знання та розуміння предметної області та розуміння професійної діяльності | ПР14. Проєктувати робочі місця виробничих підрозділів підприємств видавничо-поліграфічної галузі та організовувати їх експлуатацію з урахуванням правил охорони праці |
| ЗК-3. Здатність застосовувати знання у практичних ситуаціях | ПР14. Проєктувати робочі місця виробничих підрозділів підприємств видавничо-поліграфічної галузі та організовувати їх експлуатацію з урахуванням правил охорони праці |
| ЗК-4. Здатність приймати обґрунтовані рішення | ПР14. Проєктувати робочі місця виробничих підрозділів підприємств видавничо-поліграфічної галузі та організовувати їх експлуатацію з урахуванням правил охорони праці |
| ЗК-5. Здатність спілкуватися з представниками інших професійних груп різного рівня (з експертами з інших галузей знань/видів економічної діяльності) | ПР04. Організовувати свою діяльність для роботи автономно та в команді |
| ЗК-6. Здатність здійснення безпечної діяльності | ПР02. Знаходити, оцінювати й використовувати інформацію з різних джерел, необхідну для вирішення теоретичних і практичних завдань видавництва і поліграфії |

Закінчення табл. 1

| 1 | 2 |
|---|---|
| | ПР07. Розуміти принципи і мати навички використання технологій додрукарської підготовки, друкарських та післядрукарських процесів, теорії кольору, методів оброблення текстової та мультимедійної інформації.<br>ПР15. Оцінювати виробничі і невиробничі витрати на забезпечення виробництва продукції видавництва і поліграфії.<br>ПР20. Розробити мультимедійні продукти та їх окремі елементи.<br>ПР23. Керувати процесом виробництва друкованих та електронних видань |
| ЗК-10. Здатність зберігати та примножувати моральні, культурні, наукові цінності і досягнення суспільства на основі розуміння історії та закономірностей розвитку предметної області, її місця у загальній системі знань про природу і суспільство та у розвитку суспільства, техніки і технологій, використовувати різні види та форми рухової активності для активного відпочинку та ведення здорового способу життя | ПР02. Знаходити, оцінювати й використовувати інформацію з різних джерел, необхідну для вирішення теоретичних і практичних завдань видавництва і поліграфії.<br>ПР03. Раціонально використовувати сировинні, енергетичні та інші види ресурсів |

Для набуття наведених компетентностей, студенти мають:

**знати:**

особливості розвитку автоматизованих систем управління базами даних;

термінологію та класифікацію SQL;

технології розроблення баз даних;

вимоги та основні принципи процесу проєктування баз даних;

зміст технічного завдання на проєктування баз даних;

**вміти:**

оперувати понятійним апаратом SQL;

класифікувати бази даних та бази знань;

здійснювати обґрунтований вибір виду СУБД під потреби цільової аудиторії;

обґрунтовувати необхідність внесення певних змін у наповнення або послідовність реалізації етапів проєктування;

формувати технічне завдання на проєктування.

Завдання для самостійної роботи студентів наведено в табл. 2.

Таблиця 2

**Завдання для самостійної роботи студентів та форми її контролю**

| Назва теми | Зміст самостійної роботи студентів | Форми контролю СРС | Рекомендо-вана література | Тиждень, під час якого виконується завдання |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| **Змістовий модуль 1 Базові концепції СУБД** | | | | |
| Тема 1. Сучасні видавничі інформаційні системи | Вивчення лекційного матеріалу, підготовка до лабораторного заняття, огляд теоретичного матеріалу | Експрес-опитування | Основна: [1–4] | 1–2 |
| Тема 2. Моделі і типи даних | Вивчення лекційного матеріалу, підготовка до лабораторного заняття, огляд теоретичного матеріалу | Експрес-опитування | Основна: [1–4] | 3–4 |
| Тема 3. Реля-ційні бази даних | Вивчення лекційного матеріалу, підготовка до лабораторного заняття, огляд теоретичного матеріалу | Експрес-опитування | Основна: [1–4] | 5–6 |
| Тема 4. Проєк-тування баз даних. Нормалізація | Вивчення лекційного матеріалу, підготовка до лабораторного заняття, огляд теоретичного матеріалу | Експрес-опитування | Основна: [1–4] | 7–8 |

Закінчення табл. 2

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| **Змістовий модуль 2 Особливості програмного забезпечення систем баз даних** | | | | |
| Тема 5. Концептуальна модель бази даних. Побудова ER-діаграм | Вивчення лекційного матеріалу, підготовка до лабораторного заняття, огляд теоретичного матеріалу | Експрес-опитування | Основна: [5]. Додаткова: [6; 7]. Інформаційні ресурси: [8] | 9–10 |
| Тема 6. Мова запитів за зразком QBE | Вивчення лекційного матеріалу, підготовка до лабораторного заняття, огляд теоретичного матеріалу | Експрес-опитування | Основна: [5]. Додаткова: [6; 7]. Інформаційні ресурси: [8] | 11–12 |
| Тема 7. Мова структурованих запитів SQL | Вивчення лекційного матеріалу, підготовка до лабораторного заняття, огляд теоретичного матеріалу | Експрес-опитування | Основна: [5]. Додаткова: [6; 7]. Інформаційні ресурси: [8] | 13–14 |
| Тема 8. Технологія роботи з базами даних на платформі .NET Framework | Вивчення лекційного матеріалу, підготовка до лабораторного заняття, огляд теоретичного матеріалу | Експрес-опитування | Основна: [5]. Додаткова: [6; 7]. Інформаційні ресурси: [8] | 15–16 |

# Змістовий модуль 1
# Технологія побудови графічних об'єктів

## Тема 1. Визначення поняття інформаційної системи

### Завдання 1. Поняття "життєвий цикл інформаційної системи".

**Мета** самостійної роботи: отримання знань та навичок роботи з системами управління базами даних.

**Об'єкт** самостійної роботи – системи управління базами даних.

**Предмет** – функції адміністратора системи управління базами даних.

**Методи,** що використовуються для виконання самостійної роботи: аналіз і синтез, індукція та дедукція.

**Передбачений результат:** звіт із відповідями на запитання для самодіагностики.

*Життєвий цикл програмного забезпечення (ПЗ) – це період часу, який починається з моменту прийняття рішення про необхідність створення програмного продукту і закінчується в момент його повного вилучення з експлуатації.*

Стандарт ISO / IEC 12207: 2008 "*Information Technology – Software Life Cycle Processes*" є основним нормативним документом, який регламентує склад процесів життєвого циклу ПЗ.

Даний стандарт, використовуючи усталену термінологію, встановлює загальну структуру процесів життєвого циклу програмних засобів, на яку можна орієнтуватися в програмній індустрії. Стандарт визначає процеси, види діяльності та завдання, які використовуються під час придбання програмного продукту або послуги, а також у ході поставки, розроблення, застосування за призначенням, супроводу та припинення застосування програмних продуктів.

Кожен процес розподілений на набір дій, кожна дія – на набір завдань. Кожен процес, дія або завдання ініціюються і виконуються інших процесом у міру необхідності, причому не існує заздалегідь визначених послідовностей виконання. Зв'язки за вхідними даними при цьому зберігаються.

Кожен процес містить ряд дій. Наприклад, *процес придбання* охоплює такі дії:

- ініціація придбання;
- підготовка заявкових пропозицій;
- підготовка і коректування договору;
- нагляд за діяльністю постачальника;
- прийом і завершення робіт;

Кожна дія містить ряд завдань. Наприклад, *підготовка заявочних пропозицій* повинна передбачати такі дії:

- формування вимог до системи;
- формування списку програмного продукту;

- встановлення умов і угод;
- опис технічних обмежень (середовище функціонування системи та ін.).

Процеси життєвого циклу ПЗ розподіляють на основні, допоміжні та організаційні.

До **основних** належать такі:

**Придбання** – *дії і завдання замовника, що набуває ПЗ.*

**Поставка** – *дії і завдання постачальника, що забезпечує замовника програмним продуктом або послугою.*

**Розроблення** – *дії і завдання, що виконуються розробником: створення ПЗ, оформлення проєктної та експлуатаційної документації, підготовка тестових навчальних матеріалів та ін.*

**Експлуатація** – *дії і завдання оператора – організації, що експлуатує систему.*

**Супровід** – *дії і завдання, що виконуються організацією, яка займається супроводом, тобто службою упровадження, внесення змін у ПЗ з метою виправлення помилок, підвищення продуктивності чи адаптації до умов роботи, що змінилися або вимог.*

До **допоміжних** належать такі:

**Документування** – *формалізований опис інформації, створеної протягом ЖЦ ПЗ.*

**Управління конфігурацією** – *застосування адміністративних і технічних процедур протягом всього ЖЦ ПЗ для визначення стану компонентів ПЗ, управління його модифікаціями.*

**Забезпечення якості** – *забезпечення гарантій того, що ІС і процеси її ЖЦ відповідають заданим вимогам і затвердженим планам.*

**Верифікація** – *визначення того, що програмні продукти, які є результатами певної дії, повністю задовольняють вимоги або умови, обумовлені попередніми діями.*

**Атестація** – *визначення повноти відповідності заданих вимог і створеної системи їх конкретному функціональному призначенню.*

**Загальне оцінювання** – *оцінювання стану робіт за проєктом, контроль планування та управління ресурсами, персоналом, апаратурою, інструментальними засобами.*

**Аудит** – *визначення відповідності вимогам, планам і умовам договору.*

**Рішення проблем** – *аналіз і рішення проблем, незалежно від їх походження або джерела, виявлені під час розроблення, експлуатації, супроводу або інших процесів.*

**Організаційні** процеси містять:

**Управління** – *дії і завдання, які можуть бути виконані будь-якою стороною, яка управляє своїми процесами.*

**Створення інфраструктури** – *вибір і супровід технології, стандартів та інструментальних засобів, вибір й установка апаратних та програмних засобів, які використовуються для розроблення, експлуатації або супроводу ПЗ.*

**Удосконалення** – *оцінювання, вимірювання, контроль і вдосконалення процесів ЖЦ.*

**Навчання** – *первинне навчання та подальше постійне підвищення кваліфікації персоналу.*

**Модель життєвого циклу ПЗ** – *структура, що визначає послідовність виконання і взаємозв'язку процесів, дій і завдань протягом життєвого циклу.*

Модель життєвого циклу залежить від специфіки, масштабу і складності проєкту, специфіки умов, в яких система створюється і функціонує. На кожній стадії можуть виконуватися як кілька процесів, так і навпаки, один і той же процес може виконуватися на різних стадіях. Співвідношення між процесами і стадіями також визначається використовуваною моделлю життєвого циклу ПЗ.

За структурою розрізняють такі моделі життєвого циклу ПЗ:

- каскадна модель;
- спіральна модель;
- ітераційна модель.

*Каскадна модель* передбачає послідовне виконання всіх етапів проєкту в строго фіксованому порядку. Перехід на наступний етап означає повне завершення робіт на попередньому етапі. Вимоги, визначені на стадії формування вимог, строго документуються у вигляді технічного

завдання і фіксуються на весь час розроблення проєкту. Кожна стадія завершується випуском повного комплекту документації, достатньої для того, щоб розроблення могла бути продовжена іншою командою розробників.

*Спіральна модель* – ПЗ створюється в кілька ітерацій (витків спіралі) методом прототипування. Кожна ітерація відповідає створенню фрагмента або версії ПЗ, у ньому уточнюються цілі і характеристики проєкту, оцінюються якість отриманих результатів та плануються роботи наступної ітерації.

*Ітераційна модель* – є раціональним поєднанням моделей, що описані. Різні варіанти ітераційного підходу реалізовані в більшості сучасних технологій і методів (RUP, MSF, XP). Особливостями моделі процесів MSF є:

- підхід, заснований на фазах і віхах;
- ітеративний підхід;
- інтегрований підхід до створення та запровадження рішень.

У рамках MSF програмний код, документація, дизайн, плани та інші робочі матеріали створюються, як правило, ітеративними методами. MSF рекомендує починати розроблення вирішення з побудови, тестування і впровадження його базової функціональності. Потім до рішення додаються все нові і нові можливості. Така стратегія називається стратегією *версіонірованія*. Незважаючи на те, що для малих проєктів може бути достатнім випуск однієї версії, рекомендується не упускати можливості створення на одне рішення ряду версій. Зі створенням нових версій еволюціонує функціональність рішення.

Ітеративний підхід до процесу розроблення вимагає використання гнучкого способу ведення документації. "Живі" документи (*living documents*) повинні змінюватися в міру еволюції проєкту разом зі змінами вимог до кінцевого продукту.

У результаті виконання самостійної роботи у студента формуються компетентності: знання та розуміння предметної області та розуміння професійної діяльності.

*Завдання для самостійної роботи:*

- банк даних, компоненти, які входять до його складу;
- призначення СУБД;

- основні моделі даних;
- основні види програм, що належать до СУБД;
- зміст поняття "архітектура інформаційної системи".

### Запитання для самодіагностики

1. Дайте визначення додатку, укажіть, в яких випадках він розробляється.
2. Укажіть призначення словника даних.
3. Що становить обчислювальна система?
4. Охарактеризуйте архітектуру клієнт-сервер і назвіть варіанти її реалізації, вкажіть переваги і недоліки.
5. Зобразіть структуру інформаційної системи з файл-сервером.
6. Зобразіть структуру інформаційної системи з сервером баз даних.
7. Назвіть основні способи роботи користувача з базою даних під час вирішення прикладних задач.
8. Укажіть технології створення додатків роботи з базами даних.
9. Охарактеризуйте способи виконання додатків роботи з базами даних.
10. Розкрити зміст понять "етапи життєвого циклу" і "стадії розроблення".
11. Наведіть основні класифікаційні ознаки інформаційних систем з точки зору поширення контенту та особливостей життєвого циклу.

## Тема 2. Моделі і типи даних

### Завдання 2. Класичні та сучасні моделі подання даних.

**Мета** самостійної роботи: отримання знань та навичок роботи з сучасними моделями даних.

**Об'єкт** самостійної роботи – системи управління базами даних.

**Предмет** – функції адміністратора системи управління базами даних.

**Методи,** що використовуються для виконання самостійної роботи: аналіз і синтез, індукція та дедукція.

**Передбачений результат:** звіт із відповідями на запитання для самодіагностики.

Збережені в базі дані мають певну логічну структуру, тобто описуються деякою моделлю подання даних (моделлю даних), що підтримується СУБД. До класичних належать такі моделі даних:

- ієрархічна;
- мережева;
- реляційна;
- постреляційна;
- багатовимірна;
- об'єктно-орієнтована;
- концептуальна.

Розробляються також всілякі системи, засновані на інших моделях даних, які розширюють відомі моделі. У їх числі можна назвати об'єктно-реляційні, дидуктивно-об'єктно-орієнтовані, семантичні та інші моделі. Деякі з цих моделей слугують для інтеграції баз даних, баз знань і мов програмування.

В *ієрархічній моделі* зв'язок між даними можна описати за допомогою упорядкованого графа (або дерева). У 1968 році була введена в експлуатацію перша промислова СУБД система IMS фірми IBM. Це була перша ієрархічно база даних, завдяки якій визначили ряд фундаментальних понять у теорії систем баз даних, які і досі є основними для мережевої моделі даних.

Таку форму використовують:

- сервери каталогів, а саме LDAP і *Active Directory* (допускають чітке подання у вигляді дерева);
- файлові, база налаштувань Windows WMI і реєстр Windows;
- *Google App Engine Datastore API*;
- до переваг ієрархічної моделі даних належать ефективність використання пам'яті ЕОМ і непогані показники часу виконання основних операцій над даними. Ієрархічна модель даних зручна для роботи з ієрархічно впорядкованою інформацією;
- недоліком ієрархічної моделі є її громіздкість для оброблення інформації з досить складними логічними зв'язками, а також складність розуміння для звичайного користувача;
- на ієрархічній моделі даних засноване порівняно обмежена кількість СУБД, в числі яких можна назвати системи IMS, PC / Focus, Team-Up і Data Edge.

*Мережева модель* даних є більш загальною структурою порівняно з ієрархічною. Основні принципи мережевої моделі даних були розроблені

в середині 1960-их років, еталонний варіант мережевої моделі даних описаний у звітах робочої групи за мовами баз даних (*COnference on DAta SYstem Languages*) CODASYL (1971 р.).

У мережевих БД поряд з вертикальними реалізовані і горизонтальні зв'язки. Кожен окремий сегмент (осередок) може мати довільну кількість безпосередніх вихідних (старших) сегментів, а також довільну кількість породжених (молодших). Це забезпечує подання відносини "багато до багатьох".

Перевагою мережевий моделі даних є можливість ефективної реалізації за показниками витрат пам'яті й оперативності. Порівняно з ієрархічною моделлю, мережева модель надає великі можливості в сенсі допустимості обрання довільних зв'язків.

Недоліком мережевої моделі даних є висока складність і жорсткість схеми БД, побудованої на її основі, а також складність для розуміння і виконання оброблення інформації в БД звичайним користувачем. Крім того, в мережевий моделі даних ослаблений контроль цілісності зв'язків внаслідок допустимості встановлення довільних зв'язків між записами.

Системи на основі мережевої моделі не отримали широкого розповсюдження на практиці. Найбільш відомими мережевими СУБД є: IDMS, db_VistaIII.

Реляційна модель даних ґрунтується на понятті відношення (*relation*). Реляційна модель даних – створена Едгаром Коддом логічна модель даних, що описує:

- структури даних у вигляді (що змінюються в часі) наборів відносин;
- теоретико-множинні операції над даними: об'єднання, перетин різницю і декартове множення;
- спеціальні реляційні операції: селекція, проекція, з'єднання і поділ;
- спеціальні правила, що забезпечують цілісність даних.

Реляційна модель даних – це спосіб розгляду даних, тобто припис для способу подання даних (за допомогою таблиць) і для способу роботи з таким поданням (за допомогою операторів). Вона пов'язана з трьома аспектами даних: структурою (об'єкти), цілісністю і обробленням даних (оператори).

Перевага реляційної моделі даних полягає в простоті, зрозумілості та зручності фізичної реалізації на ЕОМ. Саме простота і зрозумілість для користувача виявилися основною причиною їх широкого використання.

Основними недоліками реляційної моделі є відсутність стандартних засобів ідентифікації окремих записів і складність опису ієрархічних і мережевих зв'язків.

Прикладами реляційних СУБД є: dBase III Plus і dBase IY (фірма *Ashton-Tate*), DB2 (IBM), R: BASE (*Microrim*), FoxPro і FoxBase (*Fox Software*), Paradox і dBASE for Windows (*Borland*), Visual FoxPro і Access (*Microsoft*), Clarion (*Clarion Software*), Ingres (*ASK Computer Systems*) і Oracle (*Oracle*).

У результаті виконання самостійної роботи у студента формуються компетентності: здатність застосовувати знання у практичних ситуаціях.

*Завдання для самостійної роботи*

- переваги і недоліки ієрархічної моделі даних;
- фізичне розміщення даних у БД ієрархічного типу;
- мережева модель даних;
- реляційна модель даних;
- переваги і недоліки постреляційної моделі;
- багатовимірна модель даних.

**Запитання для самодіагностики**

1. Назвіть і поясніть сенс операцій, здійсненних над даними в разі багатовимірної моделі.
2. Дайте визначення і наведіть приклади прояву принципів інкапсуляції, поліморфізму і наслідування стосовно до об'єктно-орієнтованих баз даних.
3. Укажіть переваги і недоліки об'єктно-орієнтованої моделі подання даних.
4. Охарактеризуйте типи даних, що використовуються в сучасних СУБД.

## Тема 3. Реляційні бази даних

### Завдання 3. Створення реляційної системи даних.

**Мета** самостійної роботи: отримання знань та навичок роботи зі створення реляційної системи даних.

**Об'єкт** самостійної роботи – системи управління базами даних.

**Предмет** – функції адміністратора системи управління базами даних.

**Методи,** що використовуються для виконання самостійної роботи: аналіз і синтез, індукція та дедукція.

**Передбачений результат:** звіт із відповідями на запитання для самодіагностики.

Як правило, будь-який вебдодаток можна розподілити на дві основні частини: фронт-енд, де відображається вся інформація сайта, і бек-енд, де дана інформація формується і розміщується (рис. 1). База даних зберігає записи в спеціально організованому вигляді, щоб інформацію можна було легко знайти і витягти. Будь-яка БД складається з однієї або декількох таблиць. Електронна таблиця складається з рядків і стовпців. Усі рядки мають однакові стовпці, а кожен стовпець містить дані.

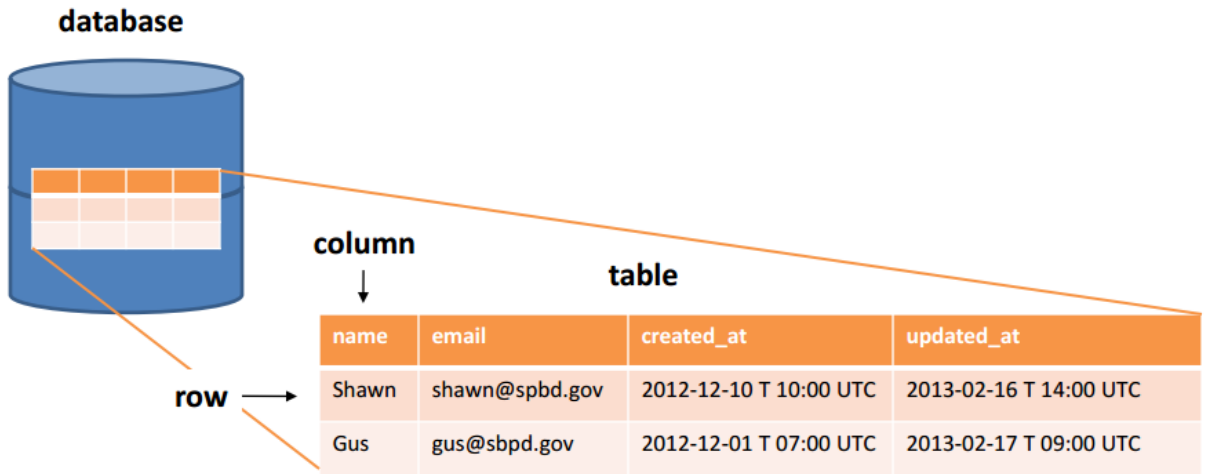


Рис. 1. **Структура бази даних**

Табличні дані можуть бути вставлені, відновлені, оновлені і видалені. Для пакета цих операцій була створена спеціальна абревіатура CRUD (*Create-Read-Update-Delete*).

Реляційні бази даних – це бази, де вся інформація зберігається в таблицях, пов'язаних один з одним спеціальними відносинами. Ці відносини дозволяють нам отримувати й об'єднувати дані з однієї або декількох таблиць за допомогою одного запиту.

Реляційна база даних – база даних, побудована на основі реляційної моделі. У реляційній базі кожен об'єкт задається відповідним записом (рядком) у таблиці. Реляційна база створюється та потім управляється за допомогою реляційної системи управління базами даних. Фактично

реляційна база даних – це інформація, що зберігається в двовимірних таблицях. Зв'язок між таблицями може знаходити своє відображення в структурі даних, а може тільки матися на увазі, тобто бути присутнім на не формалізованому рівні. Кожна таблиця БД подається як сукупність рядків і стовпців, де рядки відповідають екземпляру об'єкта, конкретної події або явища, а стовпці – атрибутам (ознаками, характеристиками, параметрами) об'єкта, події, явища. Реляційні бази даних надають більш простий доступ до складання звітів (зазвичай через SQL) і забезпечують підвищену надійність і цілісність даних завдяки відсутності надлишкової інформації.

Реляційні системи беруть свій початок у математичній теорії множин. Едгар Кодд, співробітник дослідницької лабораторії корпорації IBM, створив і описав концепцію реляційних баз даних. Нечіткість багатьох термінів, які використовуються в сфері оброблення даних, змусила Е. Кодда відмовитися від них і вигадати нові або дати більш точні визначення існуючих. Так, він не міг використовувати широко поширений термін "запис", який в різних ситуаціях може означати екземпляр запису, або тип записів, запис у стилі Кобола (що допускає повторювані групи) або плоский запис (що їх не допускає), логічний запис або фізичний запис, що зберігається, або віртуальний запис і т. д. Замість цього він використовував термін "кортеж довжини n" або просто "кортеж", якому дав точне визначення.

Е. Кодд запропонував модель, яка дозволяє розробникам розподіляти свої бази даних на окремі, але взаємопов'язані таблиці, що збільшує продуктивність, але при цьому зовнішнє подання залишається тим же, що й у вихідної бази даних. З тих пір Е. Кодда вважають батьком-засновником галузі реляційних баз даних. Е. Кодд сформулював 13 правил для реляційних баз даних, більшість яких стосуються цілісності і поновлення даних, а також доступу до них.

У результаті виконання самостійної роботи у студента формуються компетентності щодо здатності приймати обґрунтовані рішення.

*Завдання для самостійної роботи*

- сутність реляційної бази даних;
- створення ключів для зв'язку відносин;
- види зв'язків між таблицями;

- реляційні системи управління базами даних: SQLITE, MYSQL, POSTGRESQL.

**Запитання для самодіагностики**

1. Назвіть підходи до проєктування структур даних.
2. У чому полягає надмірне і не надлишкове дублювання даних?
3. Назвіть і охарактеризуйте основні види аномалій.
4. Як формується вихідне відношення під час проєктування БД?
5. Наведіть приклади явної і неявної надмірності.
6. Назвіть основні види залежностей між атрибутами відношень.
7. Наведіть приклади функціональної і часткової функціональної залежностей.

## Тема 4. Проектування баз даних. Нормалізація

**Завдання 4. Приклади відносин з залежними атрибутами.**

**Мета** самостійної роботи: отримання знань та навичок роботи з вибору структур таблиць, їх взаємозв'язки, вид нормальних форм таблиць.

**Об'єкт** самостійної роботи – системи управління базами даних.

**Предмет** – функції адміністратора системи управління базами даних.

**Методи,** що використовуються для виконання самостійної роботи: аналіз і синтез, індукція та дедукція.

**Передбачений результат:** звіт із відповідями на запитання для самодіагностики.

Проєктування інформаційних систем, що містять бази даних, здійснюється на фізичному і логічному рівнях. Рішення проблем проєктування на фізичному рівні багато в чому залежить від СУБД, що використовується. У ряді випадків користувачеві доступні налаштування окремих параметрів системи, яка не складає великої проблеми.

Логічне проєктування полягає у визначенні кількості і структури таблиць, формуванні запитів до БД, визначенні типів звітних документів, розроблення алгоритмів оброблення інформації, створення форм для введення і редагування даних у базі і вирішенні ряду інших завдань.

Вирішення завдань логічного проєктування БД в основному визначається специфікою завдань предметної області. Найбільш важливою

тут є проблема структуризації даних, на ній варто зосередити основну увагу.

Під час проєктування структур даних для автоматизованих систем можна виокремити три основні підходи:

1. Збирання інформації про об'єкти завдання, яке вирішують у рамках однієї таблиці і подальша декомпозиція його на декілька взаємопов'язаних таблиць на основі процедури нормалізації відносин.

2. Формулювання знань про систему (визначення типів вихідних даних і їх взаємозв'язків) і вимог до оброблення даних, отримання за допомогою CASE-системи (системи автоматизації проєктування і розроблення баз даних) готової схеми БД або навіть готової прикладної інформаційної системи.

Структурування інформації для використання в інформаційної системі в процесі проведення системного аналізу на основі сукупності.

Проєктування БД є одним з етапів життєвого циклу інформаційної системи. Основним завданням, що вирішується в процесі проєктування БД, є завдання нормалізації її відносин. Розглянутий далі метод нормальних форм є класичним методом проєктування реляційних БД. Цей метод заснований на фундаментальному в теорії реляційних баз даних понятті залежності між атрибутами відносин.

Процес проєктування БД з використанням методу нормальних форм є ітераційним і полягає в послідовному перекладі відносин з першої нормальної форми в нормальні форми більш високого порядку за певними правилами. Кожна наступна нормальна форма обмежує певний тип функціональних залежностей, усуває відповідні аномалії під час виконання операцій над відношеннями БД і зберігає властивості попередніх нормальних форм.

Виокремлюють таку послідовність нормальних форм:

- перша нормальна форма (1НФ);
- друга нормальна форма (2НФ);
- третя нормальна форма (3НФ);
- посилена третя нормальна форма, або нормальна форма Бойса-Кодда (БКНФ);
- четверта нормальна форма (4НФ);
- п'ята нормальна форма (5НФ).

У загальному випадку між двома атрибутами одного відношення можуть існувати залежності: 1:1, 1:М, М:1 і М:М. Оскільки залежність між

атрибутами є причиною аномалій, намагаються розчленувати відносини з залежностями атрибутів на ніскільки відносин. У результаті утворюється сукупність пов'язаних відношень (таблиць) зі зв'язками виду 1:l, 1:M, M:1 і M:M. Зв'язки між таблицями відображають залежності між атрибутами різних відносин.

У результаті виконання самостійної роботи у студента формуються компетентності: здатність спілкуватися з представниками інших професійних груп різного рівня (з експертами з інших галузей знань/видів економічної діяльності).

*Завдання для самостійної роботи*

- визначення першої нормальної форми;
- визначення другої нормальної форми;
- визначення третьої нормальної форми;
- визначення посиленої третьої нормальної форми.

**Запитання для самодіагностики**

1. Охарактеризуйте нормальні форми.
2. Назвіть рекомендації з організації зв'язку сутностей.
3. Дайте визначення фізичної та логічної цілісності БД.
4. Наведіть приклади обмежень значень і структурних обмежень.
5. Поясніть поняття зовнішнього і первинного ключів таблиць.

# Змістовий модуль 2
# Особливості програмного забезпечення систем баз даних

## Тема 5. Концептуальна модель бази даних. Побудова ER-діаграм

### Завдання 5. Основні поняття методу сутність-зв'язок.

**Мета** самостійної роботи: отримання знань та навичок роботи з діаграмами ER-екземплярів і діаграмами ER-типу.

**Об'єкт** самостійної роботи – системи управління базами даних.

**Предмет** – функції адміністратора системи управління базами даних.

**Методи,** що використовуються для виконання самостійної роботи: аналіз і синтез, індукція та дедукція.

**Передбачений результат:** звіт із відповідями на запитання для самодіагностики.

У реляційних СУБД для виконання операцій над відносинами використовуються дві групи мов, які мають у якості своєї математичної основи теоретичні мови запитів:

- реляційна алгебра;
- реляційне числення.

У *реляційній алгебрі* операнди і результати всіх дій є відносинами. Мови реляційної алгебри є процедурними, оскільки відношення, що є результатом запиту до реляційної БД, обчислюється під час виконання послідовності операторів. Оператори складаються з операндів, у ролі яких виступають відносини, і реляційних операцій. Результатом реляційної операції є відношення.

Мови *обчислень*, на відміну від реляційної алгебри, є не процедурними і дозволяють висловлювати запити за допомогою предиката першого порядку (висловлювання у вигляді функції), якому повинні задовольняти кортежі або домени відношення.

Збережені в базі дані можна обробляти вручну, послідовно переглядаючи і редагуючи дані в таблицях, за допомогою наявних у СУБД відповідних коштів. Для підвищення ефективності промінюють запити, що дозволяють виробляти множинне оброблення даних, тобто одночасно вводити, редагувати і видаляти безліч записів, а також обирати дані з таблиць.

*Запит є спеціальним чином описаною вимогою, що визначає склад зроблених над БД операцій щодо вибірки, видалення або модифікацій збережених даних.*

Для підготовки запитів за допомогою різних СУБД найчастіше використають дві основні мови опису запитів:

*QBE (Query By Example) – мова запитів за зразком;*

*SQL (Structured Query Language) – структурована мова запитів.*

У разі можливості маніпулювання даними під час опису запитів зазначені мови практично еквівалентні. Головна відмінність між ними полягає у способі формування запитів: мова QBE припускає ручне або візу-

альне формування запиту, в той час як використання SQL означає програмування запиту.

Структурована мова запитів SQL заснована на реляційному обчислюванні зі змінними кортежами. Мова має кілька стандартів, найбільш поширеними з яких є SQL-89 і SQL-92.

Мова SQL становить набір операторів. Мова SQL є алгоритмічною мовою і належить до мов, що інтерпретуються. Це означає, що кожен оператор SQL виконується безпосередньо під час його появи.

У зв'язку з цим SQL автономно не використовують, зазвичай він занурений в середовище вбудованої мови програмування СУБД (наприклад, *FoxPro СУБД Visual FoxPro, ObjectPAL СУБД Paradox, Visual Basic for Applications СУБД Access*.

Усі оператори SQL розподіляють на підмножини, які вважають підмовою SQL. У кожній підмножині використовують певні оператори, задані їх ключовими словами. Підмовою SQL є такі групи операторів.

Розрізняють два основні методи використання вбудованої мови SQL: статичний і динамічний.

За умови *статичного методу* використання мови в тексті програми є виклики функцій мови SQL, які жорстко додають в виконуваний модуль після компіляції. Зміни в функції, що викликають можуть бути на рівні окремих параметрів за допомогою змінних мови програмування.

За умови *динамічного методу* використання мови передбачають динамічну побудову викликів SQL-функцій та інтерпретація цих викликів, наприклад, звернення до даних лише у віддаленій базі, у ході виконання програми. Динамічний метод зазвичай застосовують у випадках, коли в додатку заздалегідь невідомий вид SQL-виклику і він будується в діалозі з користувачем.

Основним призначенням мови SQL є підготовка і виконання запитів. У результаті вибірки даних з однієї або декількох таблиць може бути отримано безліч записів, що називають поданням.

У результаті виконання самостійної роботи у студента формуються компетентності щодо здатності здійснення безпечної діяльності.

*Завдання для самостійної роботи*

- поняття ключа сутності;
- ступінь зв'язку між сутностями;
- етапи проєктування бази даних.

**Запитання для самодіагностики**

1. Яким може бути клас приналежності?
2. Наведіть приклад діаграми ER-екземплярів зі ступенем зв'язку між сутностями 1:1 й обов'язковим класом приналежності двох сутностей.
3. Як на діаграмах ER-типу позначають ступінь зв'язку, обов'язкову і необов'язкову участь в зв'язку екземплярів сутності?
4. Наведіть приклад діаграми ER-примірників для зв'язку типу 1:M.
5. Як здійснюється формування відносин для зв'язку 1:1?
6. Сформулюйте правило формування відносин, якщо ступінь зв'язку 1:1 і клас приналежності обох сутностей є необов'язковим.
7. Сформулюйте правило формування відносини для випадку ступеня зв'язку між сутностями 1:M (M:1) й обов'язкового класу приналежності M-зв'язкової сутності.
8. Укажіть правила формування відносин для зв'язку M:M.
9. Покажіть, що отримані в прикладі з розділу відносини знаходяться в нормальній формі Бойса-Кодда.

## Тема 6. Мова запитів за зразком QBE

**Завдання 6. Загальна характеристика теоретичних мов запитів.**

**Мета** самостійної роботи: отримання знань та навичок роботи з мовою запитів за зразком QBE.

**Об'єкт** самостійної роботи – системи управління базами даних.

**Предмет** – функції адміністратора системи управління базами даних.

**Методи,** що використовуються для виконання самостійної роботи: аналіз і синтез, індукція та дедукція.

**Передбачений результат:** звіт із відповідями на запитання для самодіагностики.

У сучасних СУБД широко використовують табличні мови запитів. Найбільш поширеною серед них є мова QBE (Query-By-Example – запит на зразок (приклад)). Мова QBE призначена для роботи в інтерактивному режимі та орієнтована на кінцевого користувача. Він реалізований у багатьох сучасних СУБД, наприклад, у dBase IV і старших версіях цієї системи, Paradox, Access. Конкретні реалізації цієї мови дещо відрізняються один від одного, але всі вони побудовані за єдиним принципом.

Підхід, втілений у мові QBE, ось у чому. У вікні формування запиту, що називається вікном Query, виокремлюють дві зони. У першій з них висвічується структура таблиці, дані з якої братимуть участь у запиті. Як вихідні запиту можуть вказуватися як таблиці з БД, а й інші запити.

У другій зоні (формі запиту табличної форми) користувач визначає умови запиту. У цій зоні користувач визначає, які поля (стовпці) беруть участь у формуванні запиту, а також умови відбору та деякі інші характеристики запиту. Наприклад, якщо користувачу необхідно отримати всі рядки із заданим значенням конкретного атрибута, то у відповідному стовпчику форми запиту вказується це значення.

У випадку формування запиту починається з вибору таблиць, яким призначений запит. Далі в другій зоні вікна *Query* задають позначки (як галочки) для вказівки полів, які включають до таблиці відповідей ANSWER. Для звуження області дії запиту в ньому вказуються умови, яким повинні відповідати вибрані дані. Коли запит сформовано, можна виконати його натисканням відповідної кнопки (наприклад, *Run Query*). СУБД обробляє запит, перетворюючи його на SQL-оператор, після виконання якого результати відображаються в таблиці ANSWER.

Є два види обчислень, які можуть виконуватися в запитах, формах і звітах: це оператори, що агрегують, які виконують операції над групою рядків, і звичайні обчислення, що зачіпають окремі поля одного або декількох пов'язаних рядків.

Набір агрегатних функцій може бути різним у різних системах. Зазвичай є такі функції: *Sum* (сума), *Min* (мінімум), *Max* (максимум), *Avg* (середнє), *Count* (підрахунок). Деякі системи містять такі додаткові статистичні функції, як відхилення, стандартне відхилення, дисперсія тощо.

Використання агрегатних функцій передбачає, що таблиця впорядкована за полем (полями), за яким ведеться агрегування. Деякі СУБД самі автоматично виконують упорядкування даних за необхідними нулями, інші – ні. В останньому випадку, якщо користувач не задасть належне впорядкування, результат буде спотвореним.

У разі використання СУБД *Paradox* значення поля в таблиці відповідей обчислюється за допомогою оператора CALC, що записується в полі праворуч від віконця. Вираз, що обчислюється, може містити елемент зразка – змінну, що набуває поточного значення поля даних. Елемент зразка позначається ім'ям, перед яким набирається символ підкреслення (символ підкреслення у запиті не відображається, а елемент зразка виділяється особливим кольором), наприклад: _a, CALC _a+2.

Результати обчислень, що виводяться у полі, не запам'ятовуються у вихідній таблиці. Замість цього обчислення знову проводяться кожного разу, коли виконується запит, тому результати завжди становлять поточний вміст бази даних. Оновити обчислені результати вручну неможливо (таблиця, що містить поле, що обчислюється, має статус "тільки для читання").

Подальші дії, які необхідно виконати, щоб з'єднати таблиці, залежать від використовуваної СУБД. Так, у деяких системах для з'єднання таблиць використовуються згадані елементи зразка. В інших СУБД використовуються візуальні способи встановлення зв'язків між таблицями: для зв'язування таблиць слід мишкою позиціонуватися на потрібному полі в головній таблиці і, не відпускаючи кнопки мишки, перемістити до нуля в підпорядкованій таблиці. На екрані з'явиться лінія, яка пов'язує таблиці. Існують інші способи встановлення зв'язків.

Теоретично можливі різні типи з'єднань таблиць. Найбільш поширеним є з'єднання, за якого в таблицю відповідей поміщаються ті з'єднані рядки, для яких значення поля зв'язку головної таблиці збігається з відповідним полем у таблиці. У наведених випадках встановлюється саме таке з'єднання. Розрізняють "ліве" і "праве" з'єднання, коли в таблицю відповідей поміщаються всі рядки з головної або підлеглої таблиці відповідно, навіть якщо для них немає зв'язаних рядків в іншій таблиці. Але не всі системи дозволяють за допомогою мови QBE реалізовувати такі з'єднання. У випадках, коли можливе задавання різних типів сполук, конкретний спосіб реалізації відрізняється у різних СУБД.

Робота з декількома таблицями в конкретних СУБД відрізняється не тільки тим, яким способом можна визначити зв'язок між таблицями. Так, наприклад, одні системи зобов'язують користувача пов'язати таблиці, які вказуються як вихідні для запиту; інші – автоматично пов'язують відкриті таблиці, але тим нулям, які система сприймає як поля зв'язку (найчастіше це поля, що мають однакові імена, тип і довжину); треті – залишають ці таблиці ізольованими, якщо користувач не вказав, як вони мають бути пов'язані, четверті – виконують декартове множення відкритих таблиць. Наприклад, у *Access*, якщо таблиці не пов'язані, то під час виконання запиту це призводить до зв'язування кожного рядка однієї таблиці з кожним рядком іншого.

Крім пошукових запитів мова QBE дозволяє виконувати й інші операції, наприклад корегування даних. Набір допустимих операцій, і навіть методи їх задавання дещо різняться у різних системах.

У мові QBE, реалізованій СУБД *Paradox*, для зміни значення поля призначено оператора CHANGETO, причому під час використання цього оператора всі поля мають бути невідзначеними. Для вставки або видалення рядків (у таблицю або з таблиці) потрібно натиснути кнопку мишки, коли курсор встановлений у першій колонці запиту на ім'я таблиці. З меню обирається команда *INSERT* або *DELETE*. Для *INSERT* у вікні Query набираються значення полів рядка, що вставляється, і натискається кнопка *Run Query*. Для *DELETE* у вікні *Query* набираються умови, яким повинні задовольняти значення полів рядків, що видаляються, і натискається кнопка *Run Query*. У процесі використання команд *INSERT* та *DELETE* усі поля мають бути невідзначеними.

Для зручності сприйняття результатів мова QBE дозволяє встановити впорядкованість даних у таблиці відповідей. Можливості задавання впорядкування розрізняються у різних СУБД: деякі системи дозволяють проводити впорядкування за довільними полями, інші вимагають, щоб поле впорядкування стояло в таблиці відповідей обов'язково першим, а якщо впорядкування ведеться за кількома полями, то щоб ці поля слідували в таблиці відповідей один за одним, порядок їх розташування у таблиці бази даних; деякі СУБД розрізняють звичайне та словникове впорядкування (коли враховується та не враховується регістр відповідно), інші – ні; у деяких системах, навіть якщо не задано жодне впорядкування, то таблиця відповідей завжди видається впорядкованою за першим полем таблиці відповідей і т. д. Запити QBE можуть бути збережені для подальшого багаторазового використання.

У результаті виконання самостійної роботи у студента формуються компетентності: здатність зберігати та примножувати моральні, культурні, наукові цінності і досягнення суспільства на основі розуміння історії та закономірностей розвитку предметної області, її місця у загальній системі знань про природу і суспільство та у розвитку суспільства, техніки і технологій; використовувати різні види та форми рухової активності для активного відпочинку та ведення здорового способу життя.

*Завдання для самостійної роботи*

- характеристика мови QBE;
- визначення поняття елемента прикладу і його використання в шаблоні запиту на вибірку;

- напрями вдосконалення мови QBE у сучасних СУБД;
- функції сучасних СУБД.

**Запитання для самодіагностики**

1. Опишіть СУБД *Microsoft Access* 2010 року.
2. Як побудувати зв'язок у реляційній СУБД *Microsoft Access* 2010?
3. Які існують способи створення таблиць у СУБД *Microsoft Access*?
4. Призначення запитів у *Microsoft Access*.
5. Як і навіщо задають умови вибору під час створення запитів?
6. Призначення параметричного запиту.
7. Які основні поняття форм у СУБД *Microsoft Access* 2010?
8. Які основні поняття звітів у СУБД *Microsoft Access* 2010?

## Тема 7. Мова структурованих запитів SQL

**Завдання 7. Створення та призначення мови запитів SQL.**

**Мета** самостійної роботи: отримання знань та навичок з роботою з базою даних у SQL.

**Об'єкт** самостійної роботи – системи управління базами даних.

**Предмет** – функції адміністратора системи управління базами даних.

**Методи,** що використовуються для виконання самостійної роботи: аналіз і синтез, індукція та дедукція.

**Передбачений результат:** звіт із відповідями на запитання для самодіагностики.

Структурована мова запитів SQL заснована на реляційному обчислюванні зі змінними кортежами. Мова має кілька стандартів, найбільш поширеними з яких є SQL-89 і SQL-92.

Мова SQL становить набір операторів. Мова SQL є алгоритмічною мовою і належить до мов, що інтерпретуються. Це означає, що кожен оператор SQL виконується безпосередньо під час його появи.

У зв'язку з цим SQL автономно не використовують, зазвичай вона занурена в середовище вбудованої мови програмування СУБД (наприклад, *FoxPro СУБД Visual FoxPro, ObjectPAL СУБД Paradox, Visual Basic for Applications СУБД Access*).

Усі безліч операторів SQL розподіляють на підмножини, які вважаються підмовою SQL. У кожній підмножині використовують певні оператори, задані їх ключовими словами. Підмовою SQL є такі групи операторів.

**Оператори визначення даних (*Data Definition Language, DDL*):**

• *CREATE* – створює об'єкт БД (саму базу, таблицю, уявлення, користувача);

• *ALTER* – змінює об'єкт;

• *DROP* – видаляє об'єкт.

**Оператори маніпуляції даними (*Data Manipulation Language, DML*):**

• *SELECT* – зчитує дані, що задовольняють заданим вимогам;

• *INSERT* – додає нові дані;

• *UPDATE* – змінює існуючі дані;

• *DELETE* – видаляє дані.

**Оператори визначення доступу до даних (*Data Control Language, DCL*):**

• *GRANT* – надає користувачеві (групі) дозвіл на визначення операції з об'єктом;

• *REVOKE* – відкликає раніше видані дозволи;

• *DENY* – задає заборону, яка має пріоритет над вирішенням.

**Оператори управління транзакціями (*Transaction Control Language, TCL*):**

• *COMMIT* – застосовує транзакцію;

• *ROLLBACK* – скасовує всі зміни, зроблені в контексті поточної транзакції;

• *SAVEPOINT* – розподіляє транзакцію на більш дрібні ділянки.

Основним призначенням мови SQL є підготовка і виконання запитів. В результаті вибірки даних з однієї, або декількох таблиць може бути отримано безліч записів, зване поданням.

*Подання за своєю сутністю є таблицею, що формується внаслідок виконання запиту.*

Можна вважати його різновидом запиту, що зберігається. За однією таблицею можна побудувати кілька подань. Саме подання описується шляхом вказівки ідентифікатора подання і запиту, який повинен бути виконаний для його отримання.

У результаті виконання самостійної роботи у студента формуються компетентності: здатність зберігати та примножувати моральні, культурні, наукові цінності і досягнення суспільства на основі розуміння історії та закономірностей розвитку предметної області, її місця у загальній системі знань про природу і суспільство та у розвитку суспільства, техніки і технологій; використовувати різні види та форми рухової активності для активного відпочинку та ведення здорового способу життя.

*Завдання для самостійної роботи*

- SQL – створення бази даних і таблиць;
- створення запитів у SQL;
- отримання підсумкових значень;
- об'єднання таблиць;
- угрупування записів і функція COUNT ();
- редагування, оновлення та видалення даних.

**Запитання для самодіагностики**

1. Які існують способи створення таблиць у *MS SQL Server*?
2. Призначення запитів у *SQL Server*.
3. Як задаються умови вибору під час створення запитів?
4. Призначення параметричного запиту.
5. Які основні поняття форм у СУБД *Microsoft SQL Server*?
6. Які основні поняття звітів у СУБД *Microsoft SQL Server*?

## Тема 8. Технологія роботи з базами даних на платформі .NET Framework

### Завдання 8. Особливості використання об'єктної моделі ADO.Net.

**Мета** самостійної роботи: отримання знань та навичок роботи з базами даних на платформі .NET Framework.

**Об'єкт** самостійної роботи – системи управління базами даних.

**Предмет** – функції адміністратора системи управління базами даних.

**Методи,** що використовуються для виконання самостійної роботи: аналіз і синтез, індукція та дедукція.

**Передбачений результат:** звіт із відповідями на запитання для самодіагностики.

Для зберігання даних використовуються різні системи управління базами даних: MS SQL Server, Oracle, MySQL і т. д. І більшість великих додатків використовують для зберігання даних ці системи управління базами даних. Однак, щоб здійснювати зв'язок між базою даних і додатком на C #, необхідний посередник. І саме таким посередником є технологія ADO.NET.

ADO.NET надає собою технологію роботи з даними, яка заснована на платформі .NET Framework. Ця технологія є набором класів, через які можна відправляти запити до баз даних, встановлювати підключення, отримувати відповідь від бази даних і виробляти ряд інших операцій.

Важливо зазначити, що систем управління баз даних може бути безліч. У своїй сутності вони можуть відрізнятися. *MS SQL Server*, наприклад, для створення запитів використовує мову *T-SQL*, а *MySQL* і *Oracle* застосовують мову *PL-SQL*. Різні системи баз даних можуть мати різні типи даних. Також можуть відрізнятися якісь інші моменти. Однак функціонал ADO.NET побудований таким чином, щоб надати розробникам уніфікований інтерфейс для роботи з самими різними СУБД.

З початку 1990-их років лідируючі позиції на ринку програмного забезпечення міцно утримує компанія *Microsoft*. Першими інтерфейсами доступу до даних, випущеними компанією *Microsoft*, були DAO і ODBC.

Інтерфейс DAO – це об'єктна модель, заснована на бібліотеці *Microsoft Jet Engine*, що використовується у *Microsoft Access*. Крім самого *Access*, технологія *Jet Engine* може бути застосована до будь-якого сховища даних, що використовує індексного-послідовний метод доступу.

Інтерфейс ODBC становить класичний інтерфейс, заснований не на об'єктній моделі, а на використанні дескрипторів API у стилі мови C. Спочатку метод доступу до даних ODBC призначався для використання корпоративними базами даних, а саме *Oracle, DB 2, Sybase* і власною базою даних *Microsoft* під назвою *SQL Server*.

З точки зору розробників реляційних баз даних інтерфейс ODBC виявився практично досконалим. І настільні, і корпоративні реляційні бази даних містили драйвери ODBC. Істотний недолік інтерфейсу ODBC полягав в тому, що в ньому не використовувалася модель COM (*Component Object Model*). Набагато більш вдалими розробками *Microsoft*, призначеними для універсального доступу до даних і заснованими на використанні моделі COM, стали OLE DB і ADO.

У технології OLE DB доступ до даних здійснюється за допомогою набору чітко визначених абстрактних об'єктів, названих котипами (*cotypes*). Котипи складаються з набору добре структурованих інтерфейсів. У інтерфейсах OLE DB поряд з користувача інтерфейсами COM повинні застосовуватися вказівні знаки, структури і масиви мови С.

Бібліотека ADO за своєю сутністю є таким собі "нашаруванням" на OLE DB, однак відрізняється від нього меншою кількістю об'єктів і тим, що кожному типу об'єктів відповідає один головний інтерфейс.

Бібліотека ADO.NET підхоплює естафету, розпочату OLE DB і ADO. Код ADO.NET працює в основному в керованому середовищі .NET. Це дозволяє значно підвищити продуктивність додатків. Бібліотеки ADO.NET можуть бути використані в будь-якій мові програмування, що підтримується платформою .NET. Завдяки численним аспектам інтеграції ADO.NET і XML, комбінація ADO.NET і стека XML дозволяє здійснювати ефективний доступ до реляційних, нереляційних та інших моделей даних.

У результаті виконання самостійної роботи у студента формуються компетентності: здатність зберігати та примножувати моральні, культурні, наукові цінності і досягнення суспільства на основі розуміння історії та закономірностей розвитку предметної області, її місця у загальній системі знань про природу і суспільство та у розвитку суспільства, техніки і технологій; використовувати різні види та форми рухової активності для активного відпочинку та ведення здорового способу життя.

*Завдання для самостійної роботи:*

- Загальні риси й особливості організації *OpenGL, DirectX, Managed DirectX, WPF, XNA Framework*.

### Запитання для самодіагностики

1. Які основні математичні поняття лежать в основі опису поворотів, відображень, трансляції і масштабування?

2. Поясніть особливості використання *XAML Graphics* для відображення об'єктів 3D.

## Рекомендована література

### Основна

1. Бакаев А. А. Методы организации и обработки баз знаний / А. А. Бакаев. – Київ : Наукова думка, 2018. – 148 с.

2. Дейт К. Дж. Введение в системы баз данных / К. Дж. Дейт. – 8-е изд. – Киев : Диалектика, 2020. – 1328 с.

3. ДСТУ 2874-94. Бази даних. Терміни та визначення. – Київ : Держстандарт України, 1995. – 32 с.

4. ДСТУ 2940-94. Системи оброблення інформації. Керування процесами оброблення даних. Терміни та визначення. – Київ : Держстандарт України, 1995. – 28 с.

5. Оньон Ф. Основы ASP.NET с примерами на C# / Ф. Оньон. – Київ : Диадектика, 2019 – 304 с.

### Додаткова

6. Microsoft Corporation. Разработка Web-приложений на Microsoft Visual Basic .NET и Microsoft Visual C# .NET : учебный курс MCAD/MCSD. – Москва : Издательско-торговый дом "Русская Редакция", 2018. – 704 с.

7. Microsoft Corporation. Разработка Web-сервисов XML и серверных компонентов курс MCAD/MCSD. – Москва : Издательско-торговый дом "Русская Редакция", на Microsoft Visual Basic .NET и Microsoft Visual C# .NET : Учебный 2018. – 576 с.

### Інформаційні ресурси

8. Вендров А. М. CASE-технологии. Современные методы и средства проектирования информационных систем [Электронный ресурс] / А. М. Вендов. – Режим доступа : http://www.citforum.ru/database/case/index.shtml.

# Зміст

НАВЧАЛЬНЕ ВИДАННЯ

# ПРОЄКТУВАННЯ БАЗ ДАНИХ ТА БАЗ ЗНАНЬ

## Методичні рекомендації до самостійної роботи студентів

**спеціальності 186 "Видавництво та поліграфія" першого (бакалаврського) рівня**

*Самостійне електронне текстове мережеве видання*

Укладач **Гордєєв** Андрій Сергійович

Відповідальний за видання *О. І. Пушкар*

Редактор *В. О. Дмитрієва*

Коректор *В. Ю. Труш*

План 2022 р. Поз. № 103 ЕВ. Обсяг 35 с.

Видавець і виготовлювач – ХНЕУ ім. С. Кузнеця, 61166, м. Харків, просп. Науки, 9-А

*Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру* ***ДК № 4853 від 20.02.2015 р.***