ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
МІНІСТЕРСТВА ОСВІТИ І НАУКИ УКРАЇНИ

Н. Г. Кеберле

# БАЗИ ДАНИХ ТА ІНФОРМАЦІЙНІ СИСТЕМИ

Методичні рекомендації до виконання курсових робіт
для студентів напряму підготовки 6.040302 –
"Інформатика"

Затверджено
вченою радою ЗНУ
Протокол № __ від __.__.2009 р.

Запоріжжя 2010

УДК 004.91:004.655/658 (076.1)
ББК з973.233-018.2я73+з988-5
К 33

Кеберле Н. Г. Бази даних та інформаційні системи : Методичні рекомендації до виконання курсових робіт з дисципліни для студентів напряму підготовки 6.040302 – "Інформатика" / Н. Г. Кеберле. — Запоріжжя : ЗНУ, 2010. — 59 с.

Методичні рекомендації до виконання курсових робіт з дисципліни “Бази даних та інформаційні системи” розраховано на студентів напряму підготовки «Інформатика», які мають навички роботи з ПЕОМ та мережами ПЕОМ, знайомі з мовами програмування, мають знання з теорії множин та математичної логіки.

У виданні містяться вимоги та рекомендації до виконання курсової роботи за дисципліною, приклади звітування.

Дане видання може бути використано для проведення індивідуальних занять зі студентами, організації самостійної роботи.

Рецензент — Єрмолаєв В.А., к.ф.-м.н., доцент кафедри інформаційних технологій

Відповідальний за випуск — Борю С.Ю., к.т.н, доцент, зав. каф. інформаційних технологій

## Зміст

## Вступ

Дані методичні рекомендації розраховані на використання під час вивчення дисципліни «Бази даних та інформаційні системи» студентами математичних спеціальностей (очного та заочного відділення), а саме, при виконанні курсової роботи.

Назва курсової роботи є спільною для всіх виконавців: «Розробка інформаційної системи для предметної області …». Кожний виконавець розробляє інформаційну систему для самостійно вибраної предметної області. Теми інформаційних систем для розробки обговорюються з викладачами дисципліни.

*Метою* курсової роботи є знайомство студентів з принципами побудови сучасних баз даних та створення інформаційних систем.

*Завданнями* курсової роботи є:

- набуття умінь з самостійного аналізу реального світу на предмет відокремлення предметної області;
- набуття умінь з інтеграції розрізненої їнформації від замовника інформаційної системи з метою створення загального опису предметної області, що буде містити опис необхідних об’єктів та операцій над ними у інформаційній системі;
- набуття умінь з проектування структури системи баз даних на базі загального опису предметної області;
- набуття і вдосконалення практичних навичок зі створення схеми системи баз даних в обранії СУБД;
- набуття і вдосконалення практичних навичок зі створення системи запитів та інтерфейсу користувачів, що відповідатиме вимогам замовника інформаційної системи.

*Середовищем розробки* курсової роботи є система управління базами даних (СУБД) Microsoft Access. При наявності потреби використання іншої (інших) середовищ розробки необхідність цього вирішується із керівником курсової роботи особисто.

*Звіт* з виконання курсової роботи складається з двох складових:

1. Інформаційна система (включаючи всі необхідні файли) на змінному носії інформації (гнучкому диску, CD диску тощо).
2. Пояснювальна записка (звіт)

*Пояснювальна записка* (звіт) повинна бути представлена в друкованому та електронному варіантах. Вимоги до друкованого варіанту пояснювальної записки наведені у додатку А.

# Постановка завдання

Вам необхідно розробити інформаційну систему для деякої предметної області, за Вашим вибором. Для цього потрібно:

1. Розробити структуру бази даних і реалізувати її в будь-якій СУБД, що доступна у ЗНУ;
2. Заповнити базу даних тестовими даними;
3. Розробити систему запитів на вибірку і модифікацію даних;
4. Розробити дружню середу для користувачів.

## Попередні відомості:

У процесі розробки передбачається виконання наступних етапів:

Етап 1. Специфікація предметної області.
Етап 2. Концептуальне проектування бази даних. Побудова ER-діаграми.
Етап 3. Конвертація ER-діаграми в реляційну схему даних.
Етап 4. Розробка SQL-скрипта для створення бази даних, внесення даних, генерація значень полів.
Етап 5. Розробка системи запитів. Оцінка швидкості виконання запитів. Використання індексів.
Етап 6. Забезпечення контролю за даними, що вводяться. Обмеження на значення.
Етап 7. Визначення прав доступу до бази даних.

Для адекватного оцінювання використовується бальна система. Виконання кожного етапу оцінюється в 50 балів. Максимальна сумарна кількість балів за виконання етапів дорівнює 350 балів.

# Етап 1. Специфікація предметної області

**Мета:** проаналізувати інформаційні вимоги, що надходять від замовника інформаційної системи і дати опис предметної області, що включать опис виявлених об'єктів і їхніх зв'язків, а також опис необхідних операцій для роботи з даними.

## Попередні відомості:

На етапі формулювання й аналізу вимог установлюються цілі і специфічні вимоги до інформаційної системи. Ці вимоги документуються у формі, доступній як кінцевому користувачу, так і розроблювачу системи. Звичайно, при цьому використовується методика інтерв'ювання персоналу різних рівнів управління і фахівців організації, що беруть участь у процесах виробництва, обслуговування й обробки даних. У результаті співбесід визначаються інформаційні потоки, що відбивають указані процеси і їхній взаємозв'язок, а також однозначне сприймання семантики інформаційної моделі. Зібрані і документовані вимоги повинні включати обмеження безпеки, надійності, а також організаційні обмеження.

Як правило, етап формулювання й аналізу вимог включає такі кроки:

1. Визначення сфери застосування інформаційної системи як у теперішньому часі, так і в майбутньому.
2. Збір інформації про використання даних.
3. Перетворення інформаційних вимог у форму, зручну для проведення аналізу.

На етапі формулювання й аналізу вимог розглядаються вимоги всіх груп користувачів, зайнятих на різних ділянках обробки і використання даних. Тому результатом цього етапу може бути *декілька* уявлень користувачів про необхідні об'єкти й операції над ними.

Розглянемо етап формулювання й аналізу вимог для одного з зовнішніх уявлень про інформаційну систему деякої торгової фірми А. Фірма А займається дрібнооптовою торгівлею продуктами харчування, має декілька складів, рознесених територіально.

**Крок 1. Визначення сфери застосування інформаційної системи.**

Нехай в одному з зовнішніх уявлень виконується опис постачань продовольчих товарів на склади фірми А. Робітникам складів необхідно виконувати такі дії, як: перевірка супровідних документів на товар, прийом товару на склад і оформлення прибуткового складського ордера, внесення запису про поставку товару в журнал товарів, видача інформації про поточну кількість товару на складі.

**Крок 2. Збір інформації про використання даних.**

Передбачається, що в одній поставці може брати участь тільки один постачальник, постачаючи декілька видів товару. Постачальник може брати участь у декількох поставках. Постачальник може поставляти декілька товарів.

Проаналізуємо інформаційні потоки, що є в даному зовнішньому уявленні.

**Вхідні дані**: кожна поставка описується за допомогою прибуткової накладної. У накладній вказується номер накладн, назва й адреса постачальника, назва товару, кількість товару, що поставляється, ціна одиниці товару, одиниці виміру товару, дата поставки), а також прізвище людини, що приймає поставку.

**Вихідні дані**: прибутковий складський ордер, що відбиває приймання товару на склад, інформація про конкретний товар, інформація про конкретного постачальника, інформація про поточне розміщення і кількість різних товарів на конкретному складі (і сумарно по всіх складах), можливо, деяка інша інформація.

**Визначення основних операцій над даними**:

У даної предметної області типові операції обробки даних можуть бути такими:

Операція 1. Оформлення нової поставки.

Операція 2. Внесення даних про нового постачальника.

Операція 3. Внесення даних про новий товар.

Операція 4. Внесення даних про всі можливі товари, що поставляються даним постачальником.

Для кожної операції створюється спеціальна таблиця (див., наприклад, Табл. 1).

Таблиця 1 – Характеристики для Операції 1.

| Назва операції | Тип **I**-інтеракт. **B**-пакетн. | Частота виконання | Використовувані об'єкти | Скільки екземплярів бере участь | Тип доступу (**R**-читання, **W**-запис) |
|---|---|---|---|---|---|
| Операція 1 | **I** | 50/день | Поставка | 1 | W |
| | | | Товар | 1 | R |
| | | | Поставщик | 1 | R |

**Крок 3. Перетворення інформаційних вимог у форму, зручну для проведення аналізу.**

Процес перетворення інформації, зібраної під час співбесід, у форму, використовувану при аналізі, містить у собі п'ять наступних кроків:

1. Укладання списку усіх елементів даних, що використовуються або створюються у предметній області.
2. Визначення виробничих задач фірми, їхніх характеристик і даних, що використовуються у них.
3. Визначення задач управління, їхніх характеристик і даних, що використовуються у них.
4. Укладання списку всіх явних і неявних правил і ліній поведінки в управлінні діяльністю фірми.
5. Укладання списку можливих майбутніх змін і шляхів їхнього впливу на діяльність фірми.

На прикладі фірми А покажемо виконання кроку 3.

1. Випишемо усі елементи даних, що використовуваються або створюються.

Частина такого списку показана в таблиці 2.

Таблиця 2 – Визначення використовуваних елементів даних.

| № | Найменування | Визначення |
|---|---|---|
| 1 | Номер постачання | Однозначно визначає кожне постачання на будь-який із складів |

| | | |
|---|---|---|
| 3 | Поставлена кількість | Кількість одиниць одного найменування товару, виконана в одне постачання. |
| 4 | Одиниця виміру | Одиниця виміру одного найменування товару, що брали участь в одному постачанні. |
| … | … | … |
| 16 | Загальна вартість товару | Загальна вартість одного найменування товару, з огляду на всі поставки і всі продажі на поточну дату |
| … | … | … |

2. Визначимо задачі, що виконуються в рамках виробничого процесу, і дані, що використовуються у них.

Виробничими задачами працівника складу будуть такі:

- приймати й відвантажувати товар, формувати супровідні документи (використовуються дані: номер видатковою або прибутковою накладною, дата, назва товару, кількість товару, ціна одиниці, одиниці виміру, П.І.Б. відповідального, номер видаткового або прибуткового складського ордера)

3. Визначимо задачі управління виробничим процесом.

Для працівника складу вони будуть такими:

- вести карти обліку по кожному складу, що відбивають рух товарів (використовуються дані: номер видатковою або прибутковою накладною, дата, назва товару, кількість товару, ціна одиниці, одиниці виміру, П.І.Б. відповідального, номер видаткового або прибуткового складського ордера)
- складати звіти про стан складу (використовуються дані: назва товару, кількість товару, ціна одиниці, одиниці виміру, Загальна вартість товару)

4. Визначимо задачі контролю й управління, необхідні в даному зовнішньому уявленні.

Задачами контролю працівника складу будуть такі:

- контролювати, щоб кількість деяких найменувань товару було не нижче визначеного мінімального запасу

- контролювати, щоб товар з обмеженим терміном збереження після настання цього терміну був знищений
- контролювати коректність заповнення всіх необхідних полів даних при оформленні ордерів
- контролювати коректність супровідних документів

5. Визначимо можливі майбутні зміни в даному зовнішньому уявленні.

Для складських працівників зміни можуть бути такими:

- додавання фасувальних, пакувальних, маркувальних робіт із товарами
- збільшення або зменшення складських площ, поява складів різних типів (наприклад, холодильників, навісів, і т.д.)

## Завдання на етап 1

(50 балів)

Визначте ту область, що Ви хочете представити в інформаційній системі. Виберіть така область, що Вам цікава і для якої Ви зможете одержати вичерпну інформацію. При побудові концептуальної моделі обмежтеся максимум 10 сутностями і 10 зв'язками.

Опишіть обрану Вами предметну область (див. опис етапу 1). Якщо в предметній області є які-небудь критичні або відносно складні аспекти, опишіть їх.

Оцінюється чіткість і повнота опису, якість таблиці основних операцій, повнота опису виробничих задач і задач управління, а також міркування про можливі майбутні зміни в предметної області, що описується .

## Форма звіту:

1. Назва предметної області та її короткий опис.
2. Кількість груп користувачів.
3. Опис зовнішнього уявлення про предметну область з точки зору кожної групи користувачів

3.1 Визначення сфери використання інформаційної системи

3.2 Визначення інформації, що буде використана даною групою користувачів

3.2.1 Вхідні дані (список)
3.2.2 Вихідні дані (список)
3.2.3 Основні операції (оформлення див. Таблицю 1)

3.3 Приведення інформації щодо зовнішнього уявлення про предметну область з точки зору даної групи користувачів.
3.3.1 перелік всіх елементів даних, що створюються або використовуються даною групою користувачів
3.3.2 перелік всіх виробничих задач з точки зору даної групи користувачів
3.3.3 перелік всіх задач управління з точки зору даної групи користувачів
3.3.4 перелік всіх задач контролю
3.3.5 перелік можливих майбутніх змін у даному зовнішньому поданні предметної області

## Литература

Основна: [1, 3-4, 6-7]
Додаткова: [1-5,7]

# Етап 2. Концептуальне проектування предметної області. Побудова ER–діаграми

**Мета:** познайомитися з методикою ER–діаграм для концептуального проектування баз даних, навчитися будувати коректні ER–діаграми.

## Попередні відомості:

### 2.1 Об'єкти та їхні властивості

Дамо визначення основних понять семантичного моделювання:

**Сутність (entity**) – збірне поняття, деяка **абстракція об'єкта**, **процесу** або **явища**, що реально існує і про який необхідно зберігати інформацію в базі даних.

У семантичному моделюванні застосовують не просто поняття "сутність", а говорять "тип сутностей":

**Тип сутностей** визначає набір об'єктів із тим самим набором властивостей.

**Екземпляр типу сутності** – конкретний об'єкт у наборі.

*Приклад 1. Якщо ми хочемо описати всіх співробітників підприємства, то всі ті загальні властивості, що властиві всім співробітникам (це може бути "ім'я", "прізвище", "посада", "зарплата"), сформують* ***тип сутності СПІВРОБІТНИК****. Тоді всі окремі співробітники є* ***екземплярами типу сутності СПІВРОБІТНИК****.*

Вибір типів сутностей залежить від мети створення бази даних. Якщо в базі даних відділу кадрів СПІВРОБІТНИК – окремий тип сутностей із своїми характеристиками, то в базі даних проектного відділу співробітник – усього лише прізвище відповідального за проект, тобто характеристика ПРОЕКТУ як типу сутностей.

Кожен тип сутності має набір властивостей (характеристик), притаманних всім екземплярам даного типу.

**Властивість (attribute, атрибут)** – поіменована характеристика типу сутностей, що приймає значення з деякої множини значень (домену).

*Приклад 2. Усі співробітники повинні мати: ім'я, прізвище, табельний_номер, дату_надходження_на_роботу, спеціальність, дату_народження. Ці характеристики є атрибутами типу сутностей СПІВРОБІТНИК.*

Кожен тип сутностей повинен відрізнятися від всіх інших. **Унікальність типу сутностей** визначається наявністю унікальних, таких, що ідентифікують тільки цей тип, властивостей. Кожен тип сутності повинен мати унікальні властивості, властиві тільки деяким об'єктам предметної області.

*Приклад 3. Розглянемо всіх співробітників навчального закладу. Очевидно, що є різні групи співробітників – викладачі, програмісти, і т.д. Для того щоб цим групам відповідали свої типи сутностей, необхідно виділити унікальні для кожної групи співробітників властивості.*
*ВИКЛАДАЧІ відрізняються від всіх інших співробітників тим, що мають такі унікальні властивості: "учена_ступінь", "звання".*
*ПРОГРАМІСТИ відрізняються такими властивостями: "спеціалізація" (бази даних, системне програмування, комп'ютерна графіка і т.д.), "мови_програмування", якою вони володіють.*

Для ідентифікації **екземпляра типу сутностей** використовуються спеціальні властивості. Це може бути одна або декілька властивостей, значення яких дозволяють **однозначно** відрізняти один екземпляр типу сутностей від іншого. Цей набір спеціальних властивостей називається **первинним ключем**.

*Приклад 4. Будь–який екземпляр СПІВРОБІТНИКА має унікальний "ідентифікаційний код податкової інспекції". Ця властивість може бути **первинним ключем**, оскільки значення такого коду ніколи не повторюються, і в однієї людини може бути тільки один такий код.*

Аналіз властивостей різних об'єктів показав, що властивості бувають:

а) одиничні і множинні

**Одинична властивість —** кожний екземпляр типу сутності має єдине значення цієї властивості.

**Множинна властивість** — кожний екземпляр типу сутності має більш одного значення цієї властивості.

*Приклад 6. Тип сутностей "ДЕТАЛЬ": властивість "назва" – одиничне: Гайка, Болт, Гвинт, Шуруп. Властивість "діаметр" – множинне: деталь Гайка (один з екземплярів ДЕТАЛІ) може бути різних діаметрів.*

б) статичні і динамічні

**Статична властивість** не змінюється з часом, на відміну від **динамічної властивості**.

*Приклад 7. Тип сутностей "ОСОБИСТІСТЬ": властивість "рік_народження" – статичне, властивість "учений_ступінь" – динамічне.*

в) обов'язкові й умовні властивості

**Властивість обов'язкова**, якщо воно повиннео бути присутнім у всіх екземплярів даного типу сутностей.
**Властивість умовна**, якщо воно може відсутньому в деяких екземплярів даного типу сутностей.

*Приклад 8. Тип сутностей "ОСОБИСТІСТЬ": властивість "учений ступінь" – умовна, тому що його присутність необов'язкова у екземплярів ОСОБИСТОСТЕЙ (таких як секретар–референт, асистент), а властивість "прізвище" – обов'язкова.*

г) складені властивості

*Приклад 9. Тип сутностей "ОСОБИСТІСТЬ": властивість "адреса" складена:*
*"адреса" = "місто" + "вулиця" + "будинок" + "квартира", де символ "+" означає "конкатенація рядків"*
*Властивість "дата_народження" = "рік" + "місяць" + "день" – теж складена.*

### 2.2 Взаємовідносини об'єктів: зв'язкі

Об'єкти, процеси, явища існують не просто самі по собі, а знаходяться у взаємозв'язку один з одним.

*Приклад 10. У предметної області "Концерти артистів естради" можна знайти такі взаємодії: є тип сутностей "СПОНСОР" (із властивостями: "прізвище", "адреса", "телефон") і тип сутностей "КОНЦЕРТ" (із властивостями: "назва_концерту", "виконавець", "кількість_номерів_у програмі", "місце_проведення"). Кожен СПОНСОР цілком може фінансувати декілька КОНЦЕРТІВ, у той же час кожен КОНЦЕРТ може бути організований одночасно декількома СПОНСОРАМИ.* ***Взаємодії*** *між двома типами сутностей подані фразами "може фінансувати декілька" і "може бути організований декількома".*

Говорять, що в предметної області об'єкти **взаємодіють** один з одним за допомогою **зв'язків (relationships)**.

У зв'язку може брати участь два і більш об'єкти. Зв'язки, у яких беруть участь два об'єкти, називаються **бінарними**. Зв'язки, у яких беруть участь три об'єкти – **тернарні**, і т.д.

Розрізняють такі типи зв'язків: "один до одного"(1:1), "один до багатьох" (1: M), "багато до багатьох" (M : N).

Зв'язок "**один до одного**" означає, що кожному екземпляру одного типу сутностей відповідає максимум 1 екземпляр іншого типу сутностей, і навпаки (див. рис.1.1)

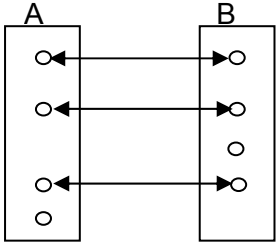



Рисунок 1.1 – Відповідність екземплярів при зв’язку 1:1.

*Приклад 11.“ОСОБИСТІСТЬ”* і*“ДНК"*
*У кожної особистості є унікальний код ДНК. Тип сутностей "ДНК" у такому випадку може мати єдину властивість: "код_ДНК"*

Зв'язок "**один до багатьох**" означає, що кожному екземпляру одного типу сутностей (A) відповідає 1 або більш екземплярів іншого типу сутностей (B), однак кожному екземпляру типу B відповідає тільки один екземпляр типу A (див. рис.1.2).

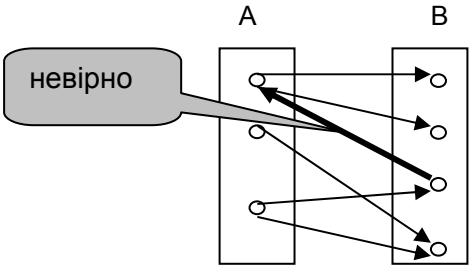



Рисунок 1.2 – Відповідність екземплярів при зв’язку 1:M

*Приклад 12. Зв'язок "1:M":*
*"КЕРІВНИК" "відповідає за декілька" “ПРОЕКТІВ”, у той час як у кожного "ПРОЕКТУ" тільки один відповідальний "КЕРІВНИК". "ВЛАСНИК" "має декілька" "АВТОМОБІЛІВ", але у кожного "АВТОМОБІЛЯ" тільки один "ВЛАСНИК". "МІСТО" "складається з декількох" "РАЙОНІВ", але кожний "РАЙОН" знаходиться тільки в одному "МІСТІ".*

Зв'язок "**багато до багатьох**" означає, що кожному екземпляру одного типу сутностей (A) відповідає 1 або більш екземплярів іншого типу сутностей (B), і навпаки (див. рис.1.3)
*Приклад 13. Зв'язок "N:M":*
*"СТУДЕНТ" "вивчає декілька" "ДИСЦИПЛІН", і навпаки "ДИСЦИПЛІНУ" "вивчають багато" " СТУДЕНТІВ"*
*"ДЕТАЛЬ" "поставляють декілька" "ПОСТАЧАЛЬНИКІВ", і навпаки кожний "ПОСТАЧАЛЬНИК" "здійснює постачання декількох" "ДЕТАЛЕЙ".*

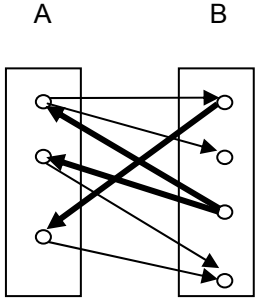



Рисунок 1.3 – Відповідність екземплярів при зв’язку N:M

Важливою характеристикою будь–якого зв'язку є її **клас приналежності**.

Клас приналежності показує, чи обов'язковий зв'язок між екземплярами цих типів сутностей чи ні.

*Приклад 14. Зв'язок обов'язковий: кожний "СТУДЕНТ" повинен прослухати декілька "ДИСЦИПЛІН" (зв'язок 1:М), обов'язкова з боку "СТУДЕНТА" і необов'язкова з боку "ДИСЦИПЛІНИ".*

*Зв'язок необов'язковий із боку "ОСОБИСТОСТІ": "ОСОБИСТІСТЬ" "може мати максимум один" "АВТОМОБІЛЬ" (зв'язок 0:1)*

*Зв'язок обов'язковий із двох боків: "ОСОБИСТІСТЬ" "повинна мати" "ДНК" (зв'язок 1:1).*

Для повного визначення зв'язку використовується поняття потужності зв'язку.

**Потужність** зв'язку – кількість екземплярів одного об'єкта, зв'язаних з одним екземпляром іншого об'єкта.

Для зв'язків визначаються **мінімальні** і **максимальні потужності**.

Якщо А і В – типи сутностей, і між ними існує зв'язок АВ, то формат специфікації зв'язку такий:

$$(m_B, M_B):(m_A, M_A)$$

де

$m_B$ - мін. кількість екземплярів В, зв'язаних з одним екземпляром А

$M_B$ - макс. кількість екземплярів В, зв'язаних з одним екземпляром А

$m_A$ - мін. кількість екземплярів А, зв'язаних з одним екземпляром В

$M_A$ - макс. кількість екземплярів А, зв'язаних з одним екземпляром В

Виконуються такі нерівності:

$$m_A \le M_A,\ m_B \le M_B,$$

значення $m_A, m_B \in \{0,1\}$, значення $M_A, M_B \in \{1, M\}$

*Приклад 15. Якщо у фірмі є правило: "кожний співробітник може брати участь принаймні в однім проекті, але максимум у 5 різних проектах", то зв'язок між "СПІВРОБІТНИКОМ" і "ПРОЕКТОМ" буде виражений як* ***"(1,5) : (1,N)"****, оскільки над кожним "ПРОЕКТОМ" може працювати від 1 до N "СПІВРОБІТНИКІВ".*

Зв'язки, так само як і типи сутностей, можуть мати властивості.

*Приклад 15. Зв'язок між "СПОНСОРОМ" і "КОНЦЕРТОМ" може мати властивості "обсяг фінансування" (скільки грошей вкладає спонсор у конкретний концерт) і "тип підтримки" (інформаційна, транспортна, загальна).*

### 2.3 «Слабкі» типи сутностей

Іноді усіх властивостей одного типу сутностей буває недостатньо для того, щоб однозначно вказувати екземпляр. Тоді використовують техніку зовнішніх ідентифікаторів.

**Слабкий тип сутностей (weak entity type)** – такий тип сутностей, первинний ключ якого складається (цілком або частково) із властивостей іншого типу сутностей.

Інакше, слабкий тип сутностей називається **залежним від інших**.

*Приклад 17. Розглянемо предметну область «Користувачі електронної пошти». Кожний користувач описується "ім'ям" і "паролем", ім'я і пароль вибираються для кожного поштового серверу, на якому цей користувач тримає поштову скриньку.*

*vasya@zsu.zp.ua, пароль gfhjkm, і vasya@mail.ru, пароль gfccdjhl*

*У свою чергу, кожний поштовий сервер характеризується властивостями "адреса поштового серверу", "розмір поштової скриньки", і т д. Це дозволяє виділити два типи сутностей, КОРИСТУВАЧ і СЕРВЕР. Однак усіх властивостей типу сутностей КОРИСТУВАЧ буде недостатньо, і необхідно використовувати властивість "адреса поштового серверу", що є ключем типу сутностей СЕРВЕР. Таким чином, КОРИСТУВАЧ буде* ***слабкою*** *сутністю.*

### 2.4 Складні типи сутностей

Нарешті, розрізняють такі типи сутностей: **прості**, тобто неподільні, і **складні**.

Складні типи сутностей бувають:

- складені – відповідають відображенню "ціле – частина";
- узагальнені – відповідає відображенню "род–вид" або "супертип–подтип"
- агреговані– відповідає звичайно якому процесу, у якому беруть участь інші об'єкти.

*Приклад 18. Складений об'єкт "КОМП'ЮТЕР": його частини "ПРОЦЕСОР", "ОЗП", "ЖОРСТКИЙ_ДИСК", "ВІДЕОКАРТА", і т.д.*

*Узагальнений об'єкт "СПІВРОБІТНИК": його підтипи – "МЕНЕДЖЕР", "ПРОГРАМІСТ", "ДИСПЕТЧЕР".*

*Агрегований об'єкт "ПОСТАВКА": у поставці беруть участь "ПОКУПЕЦЬ", "ПОСТАЧАЛЬНИК", "ТОВАР".*

## 2.5 Проведення етапу концептуального проектування системи баз даних

Опишемо тепер сам процес концептуального проектування по кроках.

**Крок 1.** Моделювання зовнішніх уявлень користувачів про предметну область.

Кожна група користувачів дає свої відомості про предметну область, такі як: вхідні дані і вихідні дані, їхні формати, допоміжні дані, необхідні в процесі роботи, і інші відомості (наприклад, вимоги безпеки).

**Крок 2.** Неформальний опис предметної області з погляду конкретної групи користувачів.

**Крок 2.1. Розбір опису з метою відокремлення типів сутностей і їхніх властивостей.**

Для будь–якого зовнішнього уявлення потрібно виділити типи сутностей, що необхідні для його опису. В окремих випадках це зробити складно, тому що інформація про предметну область може бути подана або як тип сутностей, або як властивість, або зв'язок. Тут на допомогу може прийти інтуїція і досвід розроблювача.

*Приклад 19. "ДЕТАЛЬ" і "ВИРІБ" можна представити як окремі типи сутностей, або як тип сутностей "ВИРІБ" з атрибутом "ДЕТАЛЬ".*

Для кожного обраного типу сутностей визначаються набори властивостей, і вибирається первинний ключ (або декілька альтернативних ключів).

**Крок 2.2. Розбір опису з метою виявлення зв'язків, що існують між об'єктами в предметної області і важливих для даної інформаційної системи. Визначення типів зв'язків.**

Виявляються залежності між сутностями, визначаються властивості залежностей.

**Крок 2.3. Визначення обмежень на значення властивостей, на типи зв'язків.**

Для властивостей визначається область їхніх значень, наприклад, область значень властивості "прізвище_співробітника" є текстовий рядок довжиною не більше 50 символів.

**Крок 2.4. Визначення тих основних операцій над даними, що буде виконувати конкретна група користувачів.**

Як правило, для кожної групи користувачів існує декілька типових операцій, що будуть виконуватися в процесі роботи із системою баз даних.

**Крок 2.5. Формалізація результатів кроків 1–5 і створення *зовнішньої схеми БД* для цієї групи користувачів.**

На цьому кроці створюються формальні специфікації зовнішнього уявлення, наприклад, у виді графічних діаграм (див. п.2.6).

**Крок 3. Об'єднання зовнішніх уявлень у єдине концептуальне уявлення**

При об'єднанні зовнішніх схем можуть створюватися конструкції, що є похідними стосовно конструкцій у зовнішніх схемах. Такі конструкції називаються абстракціями.

**Абстракція** – модель системи, у якій заздалегідь опущені деякі деталі.

Цілі введення абстракцій частіше усього такі:

- об'єднати розрізнені по різних зовнішніх уявленнях користувачів властивості одного типу сутностей в один тип;
- усунути різниці в сприйманні подібних об'єктів.

Залишається відзначити, що етап концептуального проектування виконується не один раз, а декілька разів, із поступовим уточненням усіх характеристик майбутньої системи баз даних. Цей етап вважається найважливішим, оскільки помилки, допущені на цьому етапі, виявляються надалі в некоректній або незручній логічній структурі бази даних, у надмірності даних, і як наслідок, у поганій фізичній організації збереження даних, у втраті продуктивності, у витратах на перепроектування структур даних і прикладних програм у складі інформаційної системи.

### 2.6. ER-діаграми

Для досягнення наочності в представленні концептуальної і зовнішніх схем бази даних було розроблені і зараз активно використовуються графічні моделі. Графічні семантичні моделі надають можливість формального і разом з тим наочного опису предметної області.

Найбільш відомою є модель графічного подання концептуальної схеми бази даних, створена *Пітером Ченом* [7].

Вона має назву "модель "Сутність-Зв'язок" (Entity-Relationship model, ER-model). Частини предметної області, що відповідають об'єктам, властивостям і зв'язкам зображуються у вигляді діаграм.

Основні графічні примітиви ER-діаграм представлені у табл. 3.

## Завдання на етап 2

(50 балів)

Спроектуйте ER-діаграму для Вашої бази даних. Вкажіть ключові атрибути, класи приналежності зв'язків і їх потужності. Якщо необхідно, відмічайте «слабкі» типи сутностей. Використовуйте єдиний стиль формування імен атрибутів і типів сутностей. Наприклад, всі типи сутностей мають імена, що починаються з великої літери, як «Ім'яТипуСутностей», а всі атрибути мають імена, що починаються з маленької літери, як «ім'яАтрибута» або «ім'я_атрибута».

Оцінюється коректність створеної ER-діаграми і її відповідність до предметної області.

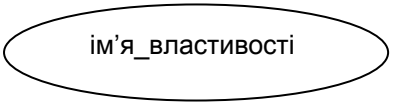
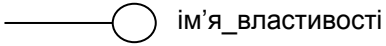
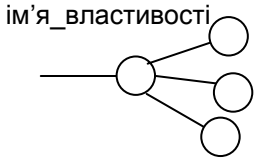
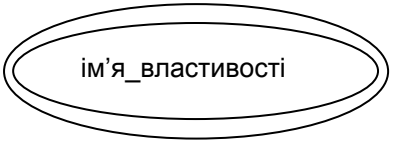
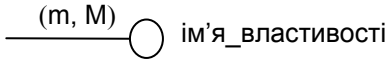
Приклад оформлення дивіться у додатку В.

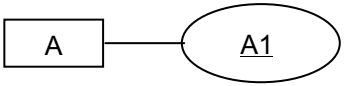
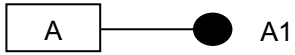
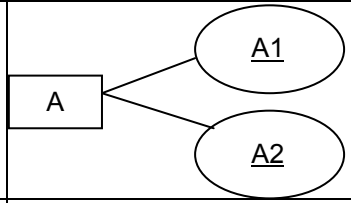
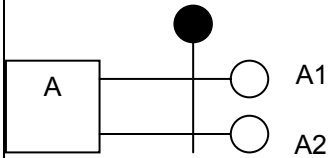
## Форма звіту:

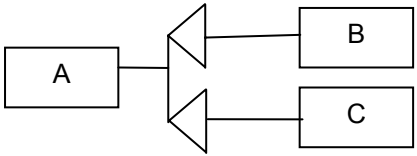
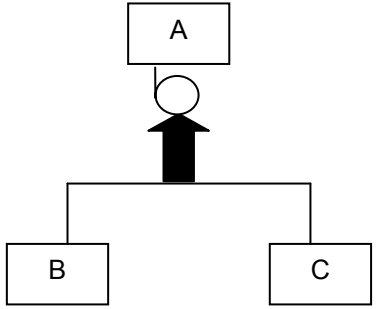
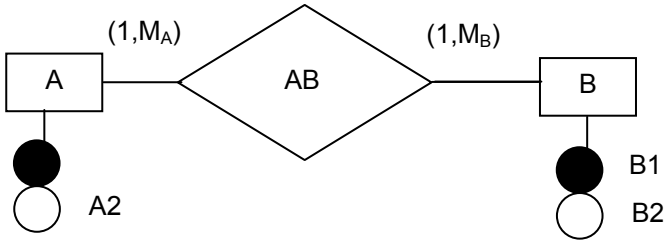
ER-діаграма, що відповідає концептуальній схемі обраної предметної області.

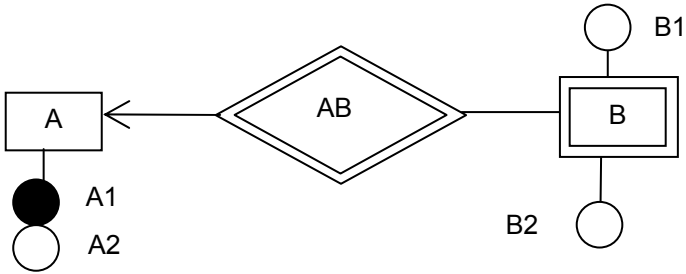
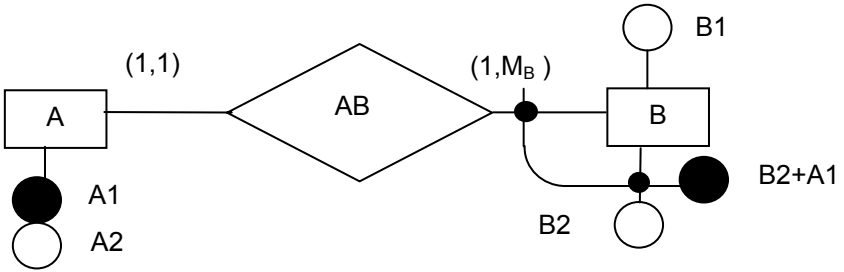
## Література

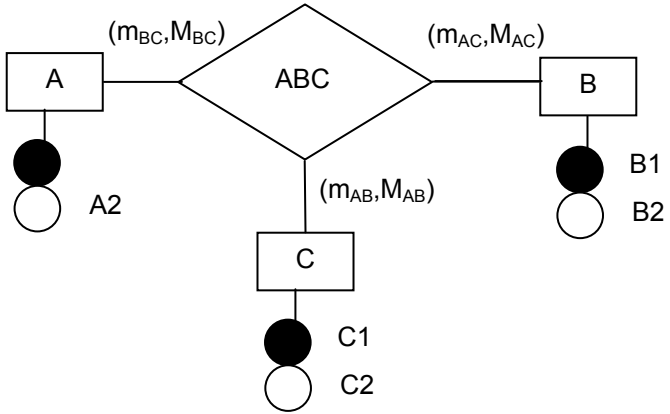
Основна: [1, 3-4, 6-7, 9]
Додаткова: [1, 3, 5-7]

| Назва примитиву | Зображення у класичній нотації Чена | Одна із сучасних нотацій |
|---|---|---|
| Тип сутностей (entity) |  |  |
| Одинична, проста властивість |  |  |
| Складена властивість |  |  |
| Множинна властивість де m - мінімальна кількість значень атрибута для одного екземпляра типу сутностей, M - максимальна кількість значень атрибута) |  |  |

| Назва примитиву | Зображення у класичній нотації Чена | Одна із сучасних нотацій |
|---|---|---|
| Первинний ключ |  |  |
| Складений первинний ключ |  |  |
| Необов'язковий зв'язок | для зв'язків з максимальними потужностями $M_B$, $M_A$ >1<br>  |  |
| Обов'язковий зв'язок | для зв'язків з максимальними потужностями $M_B$, $M_A$ =1<br>  |  |

| Назва примитиву | Зображення у класичній нотації Чена | Одна із сучасних нотацій |
|---|---|---|
| Узагальнений об’єкт (A - супертип, B,C - підтипи) |  |  |
| Складений об’єкт A (екземпляр типу сутностей A складається з більш ніж **1**, але менш ніж $\mathbf{M_B}$ екземплярів типу сутностей B) |  | |

| Назва примитиву | Зображення у класичній нотації Чена | Одна із сучасних нотацій |
|---|---|---|
| Слабкі типи сутностей (Weak entity types)<br><br>Тип сутностей **B** – слабкий, коли всіх властивостей B (наприклад, B1 і B2) недостатньо для визначення первинного ключа.<br><br>Тому говорять, що тип сутностей B ***залежить від типу сутностей*** A, і використовує первинний ключ A1 для ідентифікації екземплярів B. | В нотації Чена<br>  <br><br>В одній із сучасних нотацій<br>  <br> | |

| Назва примитиву | Зображення у класичній нотації Чена | Одна із сучасних нотацій |
|---|---|---|
| Агрегований об’єкт ABC (тип сутностей ABC представляє дію, до якої залучені типи сутностей A, B і C) |  | |

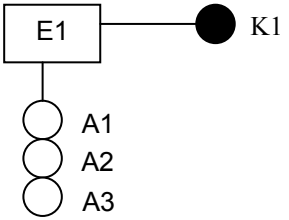
## Етап 3. Перетворення ER-діаграми в реляційну схему даних

**Мета**: навчитися виконувати конвертацію ER-діаграми в реляційну схему даних.

### Попередні відомості:

При переході від ER-діаграми до реляційної схеми даних використовують такі правила.

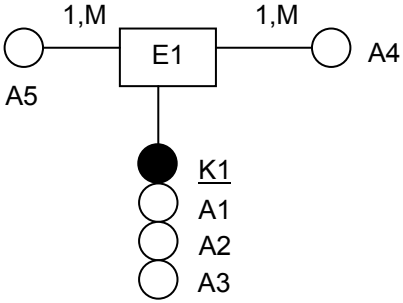
1. Для кожного простого типа сутностей і його одиничних властивостей будується реляційне відношення, атрибутами якого будуть ідентифікатор типу сутностей, а також всі одиничні властивості. Первинний ключ такого відношення – ідентифікатор типу сутностей.




**R1** (<u>K1</u>, A1, A2, A3)

Рисунок 3.1 – Тип сутностей із одиничними властивостями

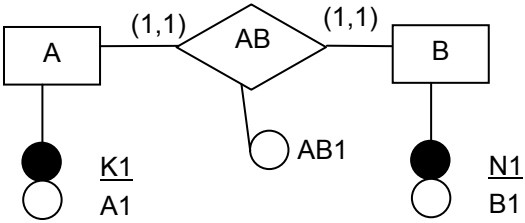
2. Якщо у типа сутностей є множинні властивості, то кожному з них ставиться у відповідність окреме відношення. Ключем цього відношення буде ідентифікатор типу сутностей, а інші властивості відповідаються даним множинним властивостям.




**R1** (<u>K1</u>, A1, A2, A3)
**R2** (<u>K1</u>, <u>A4</u>)
**R3** (<u>K1</u>, <u>A5</u>)

Рисунок 3.2 – Тип сутностей з одиничними і множинними властивостями

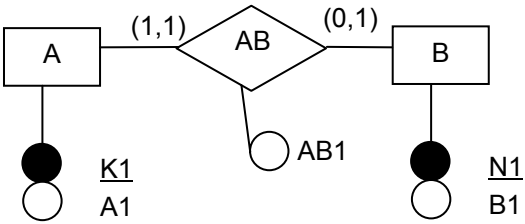
3. Якщо між двома типами сутностей є зв'язок (1,1):(1,1), тобто обов'язковий з обох сторін, то створюються два реляційних відношення, по одному на кожний тип сутностей. Первинні ключі цих відношень – відповідні ідентифікатори, і в одне з відношень додається ключ іншого, разом з усіма властивостями зв'язку (але в яке саме – не суттєво).




**R1** (K1, A1, N1, AB1)
**R2** (N1, B2)

Рисунок 3.3 – Зв'язок (1,1):(1,1) між двома типами сутностей

4. Якщо між двома типами сутностей є зв'язок (1,1):(0,1), то створюються два реляційних відношення. Кожне відношення відповідає одному типу сутностей. Первинним ключем кожного відношення є ідентифікатор відповідного типу сутностей, але первинний ключ типу сутностей, для якого мінімальна потужність зв'язку дорівнює 0, додається до відношення, що відповідає тому типу сутностей, для якого мінімальна потужність зв'язку дорівнює 1.



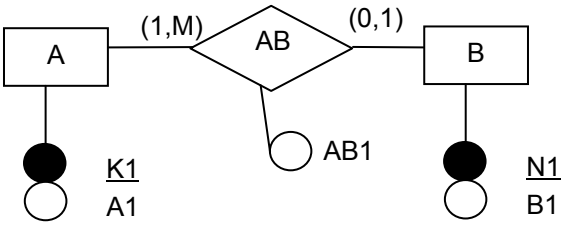

**R1** (K1, A1, N1, AB1)
**R2** (N1, B2)

Рисунок 3.4 – Зв'язок (0,1):(1,1) між двома типами сутностей.

Будемо називати **М-зв'язним типом сутностей** такий тип сутностей, для якого максимальна потужність зв'язку дорівнює М.
Будемо називати **однозв'язним типом сутностей** такий тип, для якого максимальна потужність зв'язку дорівнює 1.

5. Якщо між двома типами сутностей є зв'язок (1,M):(0,1) або (1,M):(1,1), тобто клас приналежності М-зв'язного типу сутностей є обов'язковим, то створюються два реляційних відношення. Кожне

відношення відповідає одному типу сутностей. Первинним ключем кожного відношення є ідентифікатор відповідного типу сутностей, але до однозв'язного типу сутностей додаємо ідентифікатор М-зв'язного типу сутностей.

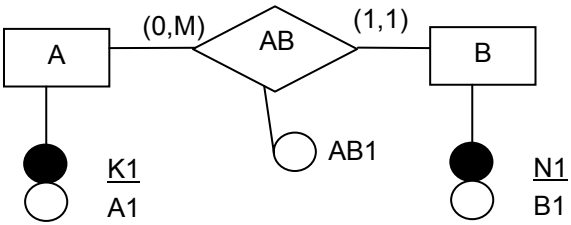



**R1** (K1, A1, N1, AB1)
**R2** (N1, B2)

Рисунок 3.5 – Зв'язки (1,M):(1,M) та (0,1):(1,M) між двома типами сутностей. Тип сутностей **B** – М-зв'язний тип сутностей.

6. Якщо між двома типами сутностей є зв'язок (0,M):(0,1), або (0,M):(1,1), або (0,M):(1,M), тобто клас приналежності хоча б одного М-зв'язного типу сутностей НЕ є обов'язковим, то створюються три реляційних відношення.

Два з них відповідають типам сутностей. Первинні ключі цих типів сутностей – відповідні ідентифікатори. Третє відношення містить ідентифікатори обох типів сутностей, а також всі властивості зв'язку.

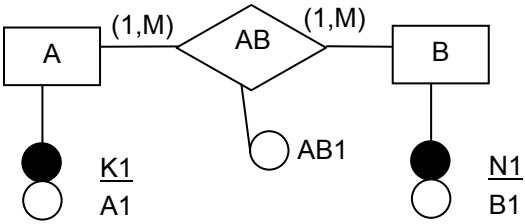



**R1** (K1, A1)
**R2** (N1, B2)
**R3** (K1, N1, AB1)

Рисунок 3.6 – Зв'язки (1,1):(0,M), (0,1):(0,M) між двома типами сутностей. Тип сутностей **B** – М-зв'язний тип сутностей.

7. Якщо між двома типами сутностей є зв'язок (0,M):(0,M) або (1,M):(1,M), то створюються три реляційних відношення.

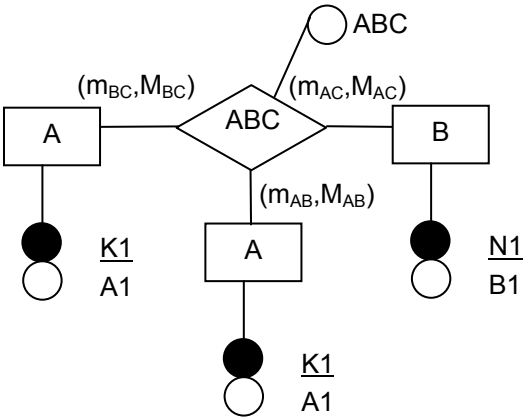
Два з них відповідають типам сутностей. Первинні ключі цих типів сутностей – відповідні ідентифікатори. Третє відношення містить ідентифікатори обох типів сутностей, а також всі властивості зв'язку.




**R1** (K1, A1)
**R2** (N1, B2)
**R3** (K1, N1, AB1)

Рисунок 3.7 – Зв'язок (1,M):(1,M) між двома типами сутностей.

8. Кожному агрегованому об'єкту відповідає окреме відношення.



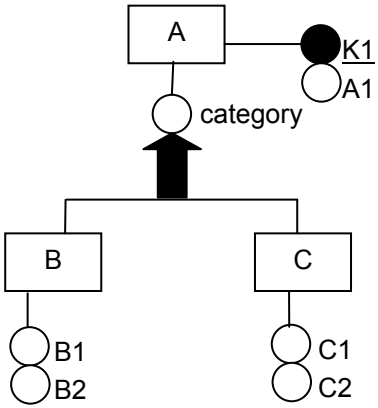

**R1** (K1, A1)
**R2** (N1, B1)
**R3** (M1, C1)
**R4** (K1, N1, M1, ABC)

Рисунок 3.8 – Агрегований об'єкт ABC, в якому беруть участь три типа сутностей.

9. Узагальнений об'єкт перетворюється таким чином (Див. рис. 3.9).

Спосіб 1. Усьому узагальненому об'єкту відповідає одне реляційне відношення (властивості об'єктів – нащадків переходять до об'єкта – батька).
Спосіб 2. Кожній категорії об'єктів – нащадків відповідає окреме реляційне відношення (властивості об'єкта – батька успадковують всі об'єкти – нащадки).

За допомогою властивості «category» відокремлюються об'єкти – нащадки один від одного.

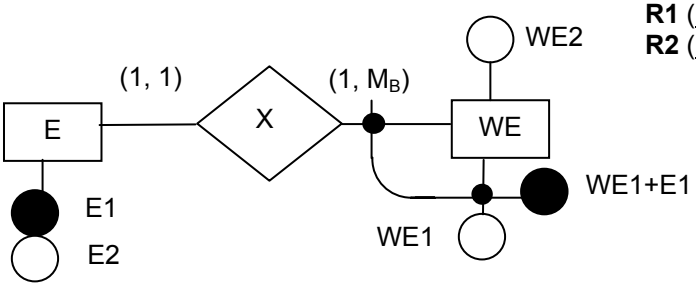



**Спосіб 1**
**R1** (K1, A1, B1, B2,
C1, C2, category)

**Спосіб 2**
**R1** (K1, A1, B1, B2)
**R2** (K1, A1, C1, C2)

Рисунок 3.9 – Способи перетворення ієрархії об'єктів.

10. Якщо між типами сутностей є зв'язок типу «ціле – частина», то цей зв'язок перетворюється згідно з класами приналежності і потужностями зв'язку, але зазвичай, як (1,M):(1,M).




**R1** (E1, E2)
**R2** (E1, WE1, WE2)

Рисунок 3.10 – «Сильний»(E) і «слабкий» (WE) типи сутностей.

11. Якщо зв'язок встановлений між «сильним» і «слабким» типами сутностей, то при перетворенні створюються два відношення, по одному на кожний тип сутностей, і первинний ключ «сильного» типу сутностей додається до ПЕРВИННОГО ключа «слабкого» типу сутностей.

## Завдання на етап 3

**(А)** (30 балів)
Використайте метод перетворення ER-діаграм в реляційні схеми (див. опис етапу 3) і створіть реляційну схему бази даних Вашої ІС.
Оцінюється коректність реляційної схеми, її відповідність до концептуальної схеми, а також до предметної області.
**(Б)** (20 балів)
З'ясуйте, чи існує можливість об'єднання відношень без виникання надлишковості. Якщо да, то які, якщо ні – то чому? Для кожної такої можливості прийміть рішення, чи треба це робити, і поясніть причини (наприклад, опишіть, для яких запитів буде зручно використовувати скореговані відношення). Доведіть, що отримана схема БД знаходиться у третій нормальній формі (3НФ).
Оцінюється змістовність Вашого обґрунтування і його відповідність до предметної області. Приклад оформлення дивіться у додатку Г.

## Форма звіту:

(А) Реляційна схема бази даних у вигляді

ІМ'Я_ВІДНОШЕННЯ (список імен атрибутів)

Ключові поля відмічені або символом «*» (зірка), або підкреслюванням. Для кожного відношення повинні бути надані визначення властивостей атрибутів (див. табл. 4).

Таблиця 4 – Характеристики атрибутів одного відношення.

| Визначення атрибута | | Властивості атрибута | | | | | |
|---|---|---|---|---|---|---|---|
| Ім'я поля | Тип даних | Розмір поля | Формат поля | Число десятк. знаків | Підпис поля | Обов'язковість поля | Є частиною індексу |

Якщо даний атрибут є частиною індексу (-ів), то вкажіть імена індексів.

(Б) Неформальне доведення того, що схема БД є у 3НФ.

## Литература

Основна: [1, 3-4, 6-7]
Додаткова: [1, 3, 6]

## Етап 4. Розробка SQL-скрипта для створення бази даних, внесення даних, генерація значень полів

**Мета:** розробити SQL-скрипт у прив'язці до конкретної СУБД, заповнити значеннями всі створені таблиці в кількості достатньої для перевірки ефективності створеної структури даних.

### Попередні відомості:

В реляційних СУБД для створення таблиць використовується частина мови SQL (Structured Query Language) під назвою DDL (Data Definition Language). SQL DDL містить конструкції для створення, видалення, а також для зміни визначення окремих стовпчиків таблиці.
Для створення таблиці використовується команда CREATE TABLE з таким синтаксисом:

```
CREATE TABLE ім'я_таблиці (
        визначення_поля[, визначення_поля,…n]
        обмеження_на_таблицю[,…n]
        );
```

де:
*визначення_поля* має вигляд
ім'я_поля тип [(розмір)] [обмеження_на_поле]
*обмеження_на_поле* має вигляд:
[CONSTRAINT ім'я_обмеження]
{PRIMARY KEY | UNIQUE | NOT NULL}

*обмеження_на_таблицю* має вигляд:
[CONSTRAINT ім'я_обмеження]
{PRIMARY KEY (ключове_поле1[, ключове_поле2 [, ...]]) |
UNIQUE (унікальне_поле1[, унікальне_поле2 [, ...]]) |
NOT NULL (непорожнє_поле1[, непорожнє_поле2 [, ...]]) |
FOREIGN KEY (поле_посилання1[, поле_посилання2 [, ...]])
REFERENCES зовнішняТаблиця [(зовнішнє_поле1 [, зовнішнє_поле2 [, ...]])]}

*ім'я_поля, ім'я_таблиці* – припустимі імена в обраній СУБД; назви таблиць та полів, *тип* – типих даних,

Для видалення таблиці використовується команда DROP TABLE:

DROP TABLE ім'я_таблиці;

Для зміни визначення окремих стовпчиків використовується команда ALTER TABLE :

ALTER TABLE *ім'я_таблиці*
{ ADD {COLUMN *ім'я_поля тип* [(*розмір*)]
[NOT NULL] [CONSTRAINT *індекс*] |
ALTER COLUMN *поле тип*[(*розмір*)]|
CONSTRAINT *ім'я_індекса*} |
DROP {COLUMN *поле* |
CONSTRAINT *ім'я_індекса* }}

## Завдання на етап 4

**(А)** (25 балів)
Запишіть SQL DDL–опис схеми бази даних для Вашої інформаційної системи. Використовуйте команди CREATE TABLE.

**(Б)** (25 балів)
Приготуйте набори значень і занесіть їх до бази даних. Якщо ці значення взяті з предметної області, для якої вже є готові набори даних, то створіть програму для їх перенесення до Вашої бази даних або імпортуйте їх із зовнішніх джерел даних.
Якщо реальних джерел інформації немає – наберіть їх вручну або створить програму їх генерації.
В будь-якому випадку, необхідно, щоб хоча б у двох реляційних відношеннях, що відповідають якимось типам сутностей або зв'язкам, було не менш ніж одна сотня записів, і хоча б у двох «словникових» відношеннях (тобто, у відношеннях, що відповідають М-зв'язаним типам сутностей) було декілька сотень записів.
При створенні програми генерації важливо, щоб:

1. Значення полів, що є первинними ключами, не дублювалися.

2. Значення первинних і зовнішніх ключів обиралися з однієї множини значень, так щоб операція натурального з'єднання повертала непусте відношення.

Додайте листинг програм переносу або генерації значень в базі даних, якщо Ви це використовували (див. Додаток В – приклад програми генерації тестових значень).

Додайте приклади кортежів, що були створені за цими програмами (не більше п'яти кортежів).

### Форма звіту:

(А)

1. Вибір СУБД та ОС.
   Викласти обґрунтування вибору СУБД та ОС.
2. SQL DDL скрипт для створення бази даних.
3. Кількість записів у кожній створеній таблиці.

(Б)

4. Фізичне зберігання даних.
   Вказати ім'я (імена) файлів бази даних, розмір файлу (-ів), імена файлів індексів (якщо індекси зберігаються окремо), розміри файлів індексів.
5. Особливості фізичного зберігання даних при промисловій експлуатації
   Вказати рішення, що були прийняті для врахування розмірів бази даних при промисловій експлуатації: планований відсоток вільного місця у файлах даних, планований відсоток вільного місця у файлах індексів, частота виконання резервного дублювання даних.
   Вказати рішення, що були прийняті для збереження OLE-, ActiveX- та інших типів об'єктів у базі даних.

### Литература

Основна: [1-6, 8]
Додаткова: [8, 9]

# Етап 5. Розробка системи запитів. Оцінка швидкості виконання запитів. Використання індексів

**Мета:** розробити систему SQL-запитів для аналізу та маніпулювання даними, в якій би виконувалися всі основні операції над даними, що були визначені на етапі аналізу предметної області. Оцінити ефективність виконання запитів і з'ясувати можливості покращення швидкості виконання запитів за допомогою системи індексів.

## Попередні відомості:

### 5.1 Конструкції SQL для аналізу та маніпулювання даними.

Інструкція SQL на **вибірку** з таблиць виглядає так:

```
SELECT [предикат] { * | таблиця.* | [таблиця.]поле_1
      [AS псевдонім_1] [, [таблиця.]поле_2 [AS псевдонім_2] [,
      ...]]}
FROM вираз [, ...] [IN зовнішняБазаДаних]
[WHERE... ]
[GROUP BY... ]
[HAVING... ]
[ORDER BY... ]
```

де:
*предикат* –ALL | DISTINCT[ROW] | [TOP n [PERCENT]]
ALL – обираються всі (в тому числі повторювані) рядки;
DISTINCT[ROW] – обираються лише різні рядки;
TOP n [PERCENT] – перші **n** записів;
TOP n PERCENT – перші **n** відсотків записів;
*таблиця* – ім'я таблиці, з якої мають бути відібрані записи;
*поле_і* — імена полів, з яких мають бути відібрані дані. Якщо включити кілька полів, вони будуть видобуватися у вказаному порядку. В якості полів також можуть бути задіяні поля, відібрані самостійним запитом на вибірку, і вирази, побудовані на основі результатів самостійних запитів;

*псевдонім_і* – імена, які стануть заголовками стовпчиків замість вихідних назв стовпчиків в таблиці;

*вираз* – імена однієї або кількох таблиць, які містять дані, що їх відбирають, або вираз, що використовує оператор INNER JOIN, LEFT JOIN, RIGHT JOIN.

Речення **FROM** має бути присутнім в кожній інструкції SELECT. Порядок слідування імен таблиць у виразі не є суттєвим.

Речення **WHERE** визначає, які записи з таблиць, перерахованих в реченні FROM, слід включити в результат виконання інструкції SELECT.

Речення **GROUP BY** об'єднує записи з однаковими значеннями у вказаному списку полів в одну групу. Якщо інструкція SELECT містить статистичну функцію SQL, наприклад, SUM() або COUNT(), то для кожної групи буде обчислено підсумкове значення.

Речення **HAVING** визначає, які згруповані записи відображаються при використанні інструкції SELECT із реченням GROUP BY. Після того як записи буде згруповано за допомогою речення GROUP BY, речення HAVING відбере ті групи записів, які відповідають умовам відбору, вказаним в реченні HAVING.

Речення **ORDER BY** сортує записи, отримані в результаті запиту, в порядку збільшення (**ASC**) чи зменшення (**DESC**) на основі значень вказаного поля чи полів.

Інструкція SQL на **додавання одного** запису має вигляд:
INSERT INTO призначення [(поле_1[, поле_2[, ...]])]
VALUES (значення_1[, значення_2[, ...])
де:

*призначення* – ім'я таблиці або запиту, в який додаються записи;

*поле_і* – ім'я поля в таблиці; можна вказувати тільки ті поля, до яких буде внесено значення, але обов'язково вказати всі ключові поля;

*значення_і* – значення, взяте з допустимих значень для *поля_і*

Для **додавання кількох** записів:
INSERT INTO призначення
[(поле_1[, поле_2[, ...]])]
SELECT [джерело.]поле_1[, поле_2[, ...]

FROM вираз
[WHERE…]
де:
*призначення* – ім'я таблиці або запиту, до якого додаються записи;
*джерело* – ім'я таблиці або запиту, звідки копіюються записи;
*поле_і* – імена полів для додавання даних, якщо вони йдуть за аргументом *призначення*; імена полів, з яких беруться дані, якщо вони йдуть за аргументом *джерело*;
*вираз* – імена таблиці або таблиць, звідки вставляються дані;
*значення _і* – значення, що додаються у вказані поля нового запису. Кожне значення буде вставлено в поле, що займає те ж положення у списку: *значення _1* вставляється в *поле_1* в новому записові, *значення _2* в *поле_2* і т.д. Кожне значення текстового поля слід заключати в лапки ' '; для розділення значень використовуйте коми.

Інструкція SQL на **створення таблиці додаванням до неї записів**:

SELECT поле_1[, поле_2[, ...]]
INTO новаТаблиця
FROM джерело
[WHERE…]
де:
*поле_і* – імена полів, які слід скопіювати в нову таблицю
*новаТаблиця* – ім'я таблиці, яку створюють. Це ім'я мусить відповідати стандартним правилам іменування. Якщо новаТаблиця збігається з іменем існуючої таблиці, виникає перехоплювана помилка.
*джерело* – ім'я існуючої таблиці, з якої відбираються записи. Це може бути: одна таблиця, кілька таблиць або запит.

Інструкція SQL на **видалення** записів має такий вигляд:

DELETE [таблиця.*]
FROM таблиця
WHERE умоваВідбору
де:
*таблиця* – ім'я таблиці, з якої видаляються записи;
*умоваВідбору* – вираз, що визначає записи, які видаляються.

Інструкція SQL на **поновлення** записів в таблиці виглядає так:

UPDATE таблиця
SET новеЗначення
WHERE умоваВідбору;

де:

*таблиця* – ім’я таблиці, дані в якій слід змінити;

*новеЗначення* – вираз, визначне значення, яке має бути вставлено у вказане поле поновлених записів.

*умоваВідбору* – вираз, що відбирає значення запису, які мають бути змінені. При виконанні цієї інструкції буде змінені тільки записи, що відповідають вказаній умові.

### 5.2 Використання індексів

Для підвищення продуктивності виконання запитів полягає у створенні індексів.

Індекс по атрибуту A реляційного відношення R дає можливість прискорити пошук кортежів з заданим значенням атрибуту A.

Індекс є корисним, якщо значення цього атрибуту часто використовується у реченні WHERE у запитах, особливо у запитах, що реалізують основні операції для роботи над даними у предметній області, а також, якщо в запитах часто виконається з’єднання по значенням атрибуту A.

Наприклад, виконання запиту

```
SELECT [Кав’ярні].[адреса]
FROM [Шанувальники_кави], [Кав’ярні]
WHERE [Шанувальники_кави].[ім’я] = 'Іван'
AND [Шанувальники_кави].[часто_провідують] = [Кав’ярні].[назва];
```

можна пришвидшити, якщо створити індекс по атрибуту [Шанувальники_кави].[ім’я], щоб швидко знайти кофе шанувальника кави Івана.

Додавши індекс [Кав’ярні].[назва] можна швидко визначити, в які кав’ярні заходить Іван, і вибрати їх адреси.

Для **створення** індексу в SQL використовується речення CREATE INDEX:

CREATE [ UNIQUE ] INDEX *ім'я_індексу*
ON *ім'я_таблиці* (*поле* [ASC|DESC][, *поле* [ASC|DESC], ...])
[WITH { PRIMARY | DISALLOW NULL | IGNORE NULL }],
де:
UNIQUE – забороняє появу значень, що дублюються, у полях індексу;
*ім'я_індексу* – ім'я створюваного індексу;
*ім'я_таблиці* – ім'я таблиці, для якої створюється індекс;
*поле* – ім'я поля, для якого створюється індекс.
PRIMARY – зробити індексовані поля первинним ключем;
DISALLOW NULL – заборонити появу NULL-значень у полях індексу;
IGNORE NULL – запобігати індексуванню записів зі значеннями NULL у полях індексу.

Для видалення індексу використовується речення

DROP INDEX *ім'я_індексу*;

## Завдання на етап 5

**(А)** (15 балів)
Розробіть запити на вибірку (використовуючи речення SELECT-FROM-WHERE), що стосуються основних операції над даними або задач управління, які були визначені на етапі 1, але не менш п'яти запитів.
Для отримання всіх балів зробіть так, щоб запити використовували всі можливості обраного Вам діалекту SQL – вибірка із декількох таблиць, підзапити, інше.
Якщо більшість запитів повертає пусті значення, то перевірте, чи виконали Ви завдання етапу 3.

**(Б)** (15 балів)
Розробіть запити на модифікацію даних (використовуючи речення INSERT, UPDATE, DELETE), що стосуються основних операції над даними або задач управління, які були визначені на етапі 1, але не менш п'яти запитів.

Для отримання всіх балів зробіть так, щоб запити використовували всі можливості обраного Вам діалекту SQL – додання в таблицю результату запиту, оновлення декількох кортежів за один раз, каскадне оновлення / видалення даних.

**(В)** (10 балів)
Створіть два представлення для бази даних (використовуючи речення CREATE VIEW).
Створіть один чи більше запитів, що звертаються до представлень.
Чи є ці представлення такими, що поновлюються? Чому?

**(Г)** (20 балів)
Створіть декілька корисних індексів, щоб індексувати поля, що найчастіше з'являються в запитах . Виконайте запити із завдання (А) з індексами та без них. Зробіть якісне порівняння швидкості виконання запитів. Приклад автоматичного додавання даних для перевірки швидкості виконання запитів дивіться у додатку Д.

## Форма звіту:

(А-В)

1. Реалізація основних операцій у інформаційній системі.
   Запити, або ланцюжки запитів, що виконують основні операції над даними, які були описані для предметної області.
   Приклади результатів запитів (надвеликі результати не друкуйте).
2. Запити, що створюють аналітичні звіти для задач управління, описаних для предметної області. Приклади результатів запитів.

(Г)

3. Індекси, що були створені для підвищення швидкості виконання основних операцій та основних аналітичних запитів.
   Швидкість роботи запитів з п.п.1-2 без індексів та з індексами.

## Література

Основна: [1-6, 8]
Додаткова: [8, 9]

# Етап 6. Забезпечення контролю за даними, що вводяться. Обмеження на значення

**Мета:** запровадити в інформаційній системі контроль за введенням/зміною даних для забезпечення цілісності бази даних.

## Попередні відомості:

При створенні інтерфейсу користувача для роботи з даними важливим є питання **контролю вводу**.
Розглядають три типи контролю:

- *Контроль типу* передбачає, що в окремий стовпчик реляційної таблиці повинні додаватися дані тільки такого типу, що був визначений при створенні таблиці. Наприклад, якщо при створенні таблиці було задано поле типу text(20), то в таке поле неможливо занести строку, чия довжина більше 20 символів.
- *Контроль діапазону* передбачає, що в окремий стовпчик реляційної таблиці повинні додаватися дані тільки з того діапазону, що був вказаний в умові на значення при створенні таблиці. Наприклад, якщо в полі «вік_співробітника» задана умова на значення «>=18 AND <=65», то в таке поле буде неможливо ввести значення, що не належить до вказаного діапазону.
- *Логічний контроль* передбачає найбільш загальний спосіб обмеження значень полів в таблицях. Логічні умови визначаються у вигляді правил (речення CREATE RULE у SQL-діалекті для MS SQL Server), якщо СУБД підтримує визначення правил. Якщо СУБД не підтримує декларативне завдання правил, то логічний контроль реалізується на рівні прикладних програм.

Проведення контролю вводу можна поєднати з інтерфейсом користувача. Серед властивостей елементів управління на формах зазвичай є такі, що забезпечують попередній контроль даних, що вводяться. При цьому формати, маски вводу та інші властивості елементів форми перевіряються у першу чергу, і лише після цього виконається перевірка форматів, макрос вводу, що визначені для даних у таблицях.

Порядок проведення контролю вводу, який треба мати на увазі при розробці інтерфейсу користувача, є таким:

- подія «до оновлення» для елемента форми

- умови на значення елемента управління
- обмеження на значення поля, що визначені у базовій таблиці (обмеження у формі логічної умови на значення поля в таблиці, обов'язковість поля, ін.)
- обмеження на значення полів, що задане для усієї таблиці

## Завдання на етап 6

(50 балів)
Розробіть систему контролю вводу/оновлення даних згідно з можливостями обраної СУБД.

Форма звіту:

1. Заходи для рішення задачи контролю цілісності реляційних даних.
   Обґрунтувати рішення щодо прийнятого рівня забезпечення цілісності даних по зв'язкам.
2. Заходи для рішення задач контролю вводу/оновлення, що були визначені для предметної області.
   Обґрунтувати рішення щодо прийнятих механізмів забезпечення контролю вводу даних.
3. Інтерфейс користувача
   Демонстрація форм, кодів обробників подій, допоміжних запитів.

## Література

Основна: [1-8]
Додаткова: [8]

## Етап 7. Визначення прав доступу до бази даних

**Мета:** розробити систему користувачів, розподілити повноваження між користувачами. Перевірити працездатність всієї інформаційної системи з урахуванням обмежень доступу.

### Попередні відомості:

Система безпеки, яку підтримує окрема СУБД, може складатися з груп користувачів з однаковими повноваженнями – такі групи називають *ролями*. Привілеї можна задавати як для ролі, так і для окремого користувача.

В стандарті SQL передбачені директиви для надання і відміни повноважень окремим користувачам.

Директива GRANT встановлює привілеї користувача.

GRANT {ALL | *привілея* [,*привілея*,…]}
ON *об'єкт*
TO {ALL | *ім'я_користувача* [,*ім'я_користувача*,…]}
[WITH GRANT OPTION];
де:

*привілея* – одна з таких:
USAGE - для використання деякого домену,
SELECT – дозволена вибірка,
INSERT – дозволено додавання записів,
UPDATE – дозволено оновлення записів,
DELETE – дозволено видалення записів,
REFERENCES – дозволено звертання в обмеженнях цілісності до відповідних об'єктів (якщо звертаються до батьківської таблиці, то з привілеєю REFERENCES можна звертатися до дочірніх таблиць);
*об'єкт* – домен чи відношення;
WITH GRANT OPTION – надає користувачеві привілею надання привілей іншим користувачам.

Директива REVOKE відкликає привілеї, що видані користувачем А користувачеві В.

REVOKE [GRANT OPTION FOR]
{ALL | *привілея* [,*привілея*,…]}
ON *об'єкт*
FROM {ALL | *ім'я_користувача* [,*ім'я_користувача*,…]} *option*;
де:

*option* – одна з таких:
CASCADE – відміна всіх привілеїв, що походять від даної привілеї
RESTRICT – заборона на відміну привілеї, якщо від неї походять інші привілеї.

## Завдання на етап 7

(50 балів)
Розробіть систему користувачів, розподіліть привілеї користувачам, згідно з можливостями обраної СУБД.

## Форма звіту:

1. Опишіть заходи, що були використані для рішення задачі розподілу прав доступу до даних, з урахуванням можливостей СУБД. Обґрунтуйте рішення щодо прийнятих прав доступу забезпечення цілісності даних по зв'язкам. Чи відповідає даний розподіл привілеїв тим групам користувачів, що були визначені під час аналізу предметної області?
2. Напишіть декілька директив (не більше п'яти) на надання привілеїв, згідно з тим, як це було зроблено у Вашій інформаційній системі.

## Литература

Основна: [1-8]
Додаткова: [1, 8]

## Термінологічний словник

**Агрегована функція** – це функція, що обчислює агреговане значення для всієї таблиці або для груп, всередині таблиці. До агрегованих функції належать: SUM (обчислення суми значень у полі), COUNT (обчислення кількості значень у полі), AVG (обчислення середнього), MIN, MAX (обчислення мінімального, максимального значення), інші.

**База даних (БД)** – це сукупність взаємопов'язаних даних, що зберігаються разом, з тією мінімальною надлишковістю, що дозволяє використовувати ці дані в кількох додатках.

**Вкладений запит** – це запит, в якому всередині речення WHERE вміщено інший запит.

**Група (або роль)** – іменований набір привілеїв для роботи у БД.

**Запис** – це окремий рядок реляційної таблиці (**кортеж** в реляційній теорії).

**Запит** – це інструкція SQL на створення об'єкта бази даних, на відбір даних з однієї чи декількох таблиць, або на зміну даних у таблиці.

**Збережена процедура** (процедура, що зберігається на сервері) - це іменований набір команд Transact-SQL, що є окремим об'єктом бази даних і зберігається на сервері СУБД.

**Зовнішній ключ** – це поле або група полів однієї таблиці (така таблиця має назву **дочірня**), яким відповідає поле або група полів, що складає первинний ключ деякої іншої таблиці (така таблиця має назву **батьківська**).

**Індекс** – об'єкт бази даних, що прискорює пошук і сортування даних.

**Інструкція SQL** – це проста або складена завершена мовна конструкція SQL. До інструкцій належать:

SELECT...FROM...WHERE, CREATE TABLE, CREATE TRIGGER, інші.

**Користувач** – людина або програма, що може виконувати певні дії у базі даних.

**Параметричний запит** – запит, обчислення якого залежить від введеного користувачем значення параметра.

**Первинний ключ** – набір полів (від одного до всіх полів), таких, що задаючи значення цьому набору полів можна повністю ідентифікувати весь запис, і притому єдиний, тобто в будь-яких двох записів реляційної таблиці значення первинного ключа не можуть збігатися.

**Пов'язаний підзапит** – це підзапит, в якому є посилання на таблицю, ім'я якої вказано в реченні FROM зовнішнього запиту. Пов'язаний підзапит виконується по одному разу для кожного рядка таблиці із зовнішнього запиту.

**Поле** – це окремий стовпчик реляційної таблиці (**атрибут** в реляційній теорії).

**Привілея** – можливість виконувати конкретну дію у базі даних.

**Протокол ODBC** (Open Database Connectivity) – стандартний протокол доступу до даних на серверах баз даних SQL, наприклад, на серверах Microsoft SQL Server або ORACLE.

**Реляційні бази даних** – це бази даних, в яких всі дані подані у вигляді двомірних таблиць особливого виду, що називаються реляційними.

**Реляційні таблиці** відрізняються від довільних двомірних таблиць за такими ознаками:

- рядки не нумеруються, порядок їх слідування не має значення;
- стовпчики не нумеруються, однозначно визначаються іменами, в межах однієї таблиці всі імена стовпчиків – унікальні;

- на перетині будь-якого стовпчика і будь-якого рядка знаходиться атомарне (неподільне) значення;
- кожна таблиця має первинний ключ.

**Речення SQL** – це частина складеної мовної конструкції SQL. До речень належать: SELECT…, FROM…, WHERE,…, HAVING…, GROUP BY…, ORDER BY…, інші.

**Розподілені системи баз даних** – це системи, в яких окремі файли даних є фізично розподіленими, і можуть знаходитися на одному й тому ж локальному комп'ютері, на різних комп'ютерах в локальній мережі, на віддалених комп'ютерах (наприклад, пов'язаних мережею на зразок Інтернет). Розрізняють **файл-серверні**, **клієнт-серверні**, **три-** і **багатоланкові** додатки для роботи з розподіленими СБД.

**Система баз даних (СБД) –** один чи декілька файлів даних, що зберігаються на одному чи декількох накопичувачах (можливо, рознесених теріторіально), що дозволяють роботу з даними одному чи декільком користувачам (можливо, одночасну).

**Система управління базами даних (СУБД)** – це пакет прикладних програм спеціального призначення для створення та робот з базою даних.

**Тригер** - це спеціальний тип збереженої процедури, що запускається сервером СУБД автоматично при виконанні з таблицею дій, що призводять до змін даних.

**Фільтр на дані** (MS Access) – це набір умов, які застосовуються для відбору підмножини записів або для сортування записів з однієї таблиці.

**Цілісність даних** - система правил, використовуваних в СУБД для підтримки зв'язків між записами в зв'язаних таблицях, а також забезпечує захист від випадкового видалення або зміни пов'язаних даних.

**SQL (Structured Query Language)** – стандартна мова запитів у реляційних базах даних.

# Література

**Основна:**


1. Гарсиа-Молина Г. Системы баз данных (полный курс) / Г. Гарсиа-Молина, Дж. Ульман, Дж. Видом. — Москва : Вильямс, 2004. — 1088, [1] с.
2. Грабер М. Введение в SQL / М. Грабер. — Москва : Лори, 1996. — 379, [1] с.
3. Дейт К. Введение в системы баз данных / К. Дейт. — 6-е изд., перераб., и доп. — Киев : Диалектика, 1998. — 784, [1] с.
4. Карпова Т. С. Базы данных: Модели, разработка, реализация / Т. С. Карпова. — Санкт-Петербург : ПИТЕР, 2001. — 304, [1] с. — (Учебник).
5. Кеберле Н. Г. Навчально-методичний посібник з дисципліни «Бази даних та інформаційні системи» для студентів математичних спеціальностей / Н. Г. Кеберле. — Запоріжжя : ЗНУ, 2010. — 105,[1] с.
6. Конолли Т. Базы данных: проектирование, реализация и сопровождение. Теория и практика / Т. Конолли, К. Бегг, А. Страчан. — Москва : Вильямс, 2000. — 1436, [1] с.
7. Райордан Р. М. Основы реляционных баз данных / Р. М. Райордан. — Москва : Русская редакция, 2001. — 384, [1] с. — (Серия «Базовый курс: теория и практика»).
8. Райордан Р. М. Программирование в Microsoft SQL Server 2000. Шаг за шагом / Р. М. Райордан. — Москва : ЭКОМ, 2002. — 608, [1] с.
9. Чен П. Модель "сущность-связь" – шаг к единому представлению о данных / П. П.-Ш. Чен // СУБД. — 1995. — №3. — С. 137—158.


**Додаткова:**


1. Атре Ш. Структурный подход к организации баз данных / Ш. Атре. — Москва : Финансы и статистика, 1983. — 320, [1] с.
2. Бойко В. В. Проектирование баз данных информационных систем / В. В. Бойко, В. Н. Савинков. — Москва : Финансы и статистика, 1989. — 351, [1] с.
3. Диго С. М. Проектирование и использование баз данных /С. М. Диго. — Москва : Финансы и статистика, 1995. — 208, [1] с.
4. Мартин Дж. Организация баз данных в вычислительных системах / Дж. Мартин. — Москва : Мир, 1980. — 662, [1] с.

5. Тиори Т. Проектирование структур баз данных : в 2 т. / Т. Тиори, Дж. Фрай. — Москва : Мир, 1985. Кн. 1. — 287, [1] с.; Кн. 2. — 320, [1] с.
6. Цикритзис Д. Модели данных / Д. Цикритзис, Ф. Лоховски. — Москва : Финансы и статистика, 1985. — 344, [1] с.
7. Ульман Дж. Основы систем баз данных / Дж. Ульман. — Москва : Финансы и статистика, 1983. — 334, [1] с.
8. Шкарина Л. Язык SQL / Л. Шкарина. — Санкт-Петербург : ПИТЕР, 2001. – 592, [1] с.
9. Пасічник В. В. Організація баз даних та знань : підруч. [для студ. виш. навч. закл.] / В. В. Пасічник, В.А. Резніченко. — К. : Видавнича група BHV, 2006. — 384, [1] с.

# Додаток А

## Основні вимоги до друкованого варіанту пояснювальної записки

1. Пояснювальна записка друкується на листах формату А4 з однієї сторони.
2. Параметри сторінки:
    - Ліве поле — 2.5 сантиметри.
    - Праве поле — 1.5 сантиметри.
    - Верхнє поле — 2.0 сантиметри.
    - Нижнє поле — 2.0 сантиметри.
3. Шрифтове оформлення:
    - Гарнітура — Times New Roman.
    - Кегль — 14 пунктів.
4. Абзацне оформлення:
    - міжрядковий інтервал — одинарний.
    - абзацний відступ — 6 пунктів.
    - вирівнювання — з обох сторін (по ширині).
5. Кожний розділ повинен починатися з нової сторінки.
6. Інтервал після назви розділу — 6 пунктів.
7. Шрифтове оформлення назви розділу:
    - Гарнітура — Times New Roman
    - Кегль — 16 пунктів.
    - Зображення — напівжирне.
8. Перед початком параграфу в межах розділу розрив сторінки робити не треба.
9. Перед назвою параграфу розділу інтервал 6 пунктів.
10. Шрифтове оформлення назви параграфу:
    - Гарнітура — Times New Roman.
    - Кегль — 14 пунктів.
    - Зображення — напівжирне.
11. Шрифтове оформлення назви пунктів в межах параграфу — довільне.
12. Вирівнювання усіх назв (розділів, параграфів, пунктів) — за центром.
13. Наприкінці усіх заголовків крапки бути не повинно.
14. Додатки нумеруються буквами українського (чи російського) алфавіту.

15. Розділи (крім вступу та заключення) нумеруються арабськими цифрами.
16. Номери сторінок повинні знаходитись у правому верхньому куті).
17. Перша сторінка (титульний лист) не повинен містити номеру.
18. Виноски повинні бути сторінкові. Їх використання не рекомендується.
19. Усі ілюстрації та їх назви повинні мати вирівнювання за центром та мати підпис виду «Рис. 1 Назва»
20. Усі назви таблиць повинні мати вирівнювання за правим полем та мати підпис виду «Таблиця 1. Назва»

*Електронний* варіант пояснювальної записки необхідно надати в *форматах* MS Word (версія не нижче 97) або PDF. Якщо пояснювальна записка надається в форматі MS Word, то її оформлення повинно зберігатися незалежно від комп'ютеру, на якому документ буде використовуватися (тобто для кожного типу абзаців необхідно визначити відповідний стиль, який зберегти у документі).

Пояснювальна записка складається з таких основних частин:

1. Титульний лист (див. додаток Б)
2. Реферат.
3. Зміст.
4. Вступ.
5. Опис етапів проектування, згідно зі змістом кожного етапу.
6. Приклади інтерфейсних рішень (форм, оброблювачів подій тощо).
7. Висновок.
8. Список використаних джерел.

*Реферат* не перевищує 1 сторінки і містить інформацію про об'єм пояснювальної записки, кількості ілюстрацій, таблиць, додатків та використаних літературних джерел. Крім цього, реферат повинен містити відомості про об'єкт дослідження, ціль дослідження, мету дослідження, результати та виводи, а також список ключових слів.

*Зміст* починається з нової сторінки після реферату і містить перелік всіх розділів пояснювальної записки. Зміст необхідно створювати автоматично, використовуючи відповідні можливості текстового процесору.

*Вступ* складається з декількох абзаців, у яких вказано предметну область для створення інформаційної системи а також середовище її розробки.

*Опис етапів проектування* складається з семи розділів, у яких, згідно із завданнями , вказаними наприкінці кожного етапу, розкривається зміст робот на цьому етапі, обгрунтування прийнятих рішень щодо структури даних.

*Приклади інтерфейсних рішень* містяться у одному чи декількох розділах, що безпосередньо описують особливості створення інтерфейсів інформаційної системи.

У *висновку* необхідно указати (коротко) назву інформаційної системи (тобто предметну область, для якої вона була створена) та основні її функціональні можливості. Якщо можливо, виділити якості, завдяки яким створена інформаційна система краща за інші подібні (для тієї ж предметної області).

*Список використаних літературних джерел* створюється за загальними правилами, які перелічені у ДСТУ ГОСТ 7.1:2006 «Система стандартів з інформації, бібліотечної та видавничої справи. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання».

*Додатки* (якщо вони є) можуть містити програмний код, блок-схеми, концептуальні схеми, та інше.

# Додаток Б

**Приклад оформлення титульного аркушу пояснювальної записки**

ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
МІНІСТЕРСТВА ОСВІТИ І НАУКИ УКРАЇНИ

Кафедра інформаційних технологій

КУРСОВА РОБОТА
з дисципліни «Бази даних та інформаційні системи»
за темою «Розробка інформаційної системи для предметної області ...»

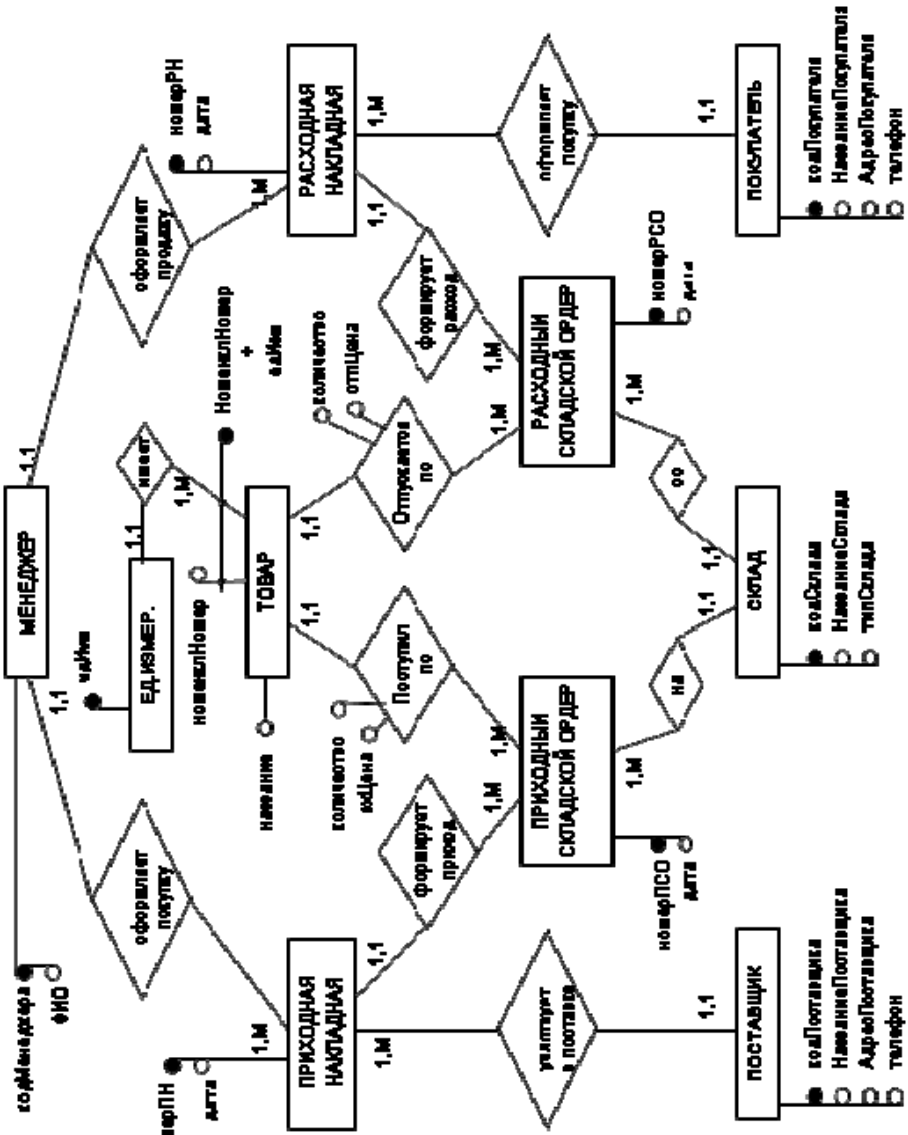
Виконав: студент групи ХХХХ-ХХ
Іванов І.І.

Прийняв:

Запоріжжя 20__

# Додаток В

## Приклад оформлення етапу 2



# Додаток Г
## Приклад оформлення етапу 3

(А)
МЕНЕДЖЕР (кодМенеджера, ФИО)
ПОСТАВЩИК (кодПоставщика, названиеПоставщика, адресПоставщика, телефон)
ПОКУПАТЕЛЬ (кодПокупателя, названиеПокупателя, адресПокупателя, телефон)
СКЛАД (кодСклада, названиеСклада, типСклада)
ЕДИЗМЕР (едИзм)
ТОВАР (номенклНомер, едИзм, название)
ПРИХОДНАЯ_НАКЛАДНАЯ (номерПН, дата, кодПоставщика, кодМенеджера)
РАСХОДНАЯ_НАКЛАДНАЯ (номерРН, дата, кодПокупателя, кодМенеджера)
ПРИХОДНЫЙ_СКЛАДСКОЙ_ОРДЕР (номерПСО, дата, номерПН, кодСклада, номенклНомер, едИзм, количество, вхЦена)
РАСХОДНЫЙ_СКЛАДСКОЙ_ОРДЕР (номерРСО, дата, номерРН, кодСклада, номенклНомер, едИзм, количество, отпЦена)

(Б) Дана схема знаходиться у 3НФ тому, що всі транзитивні і часткові залежності є залежностями від первинних ключів відповідних таблиць.

# Додаток Д
## Генерація значень в таблиці

Нехай таблиця А була створена за допомогою запиту:

```
CREATE TABLE A
( a1 autoincrement not null,
a2 text(50) not null,
a3 longint
CONSTRAINT pk_A PRIMARY KEY (a1));
```

Для генерації значень в таблиці А можна створити таку форму, як на Рис.В.1. Обробка події «нажатие кнопки» виконується функцією random_write().

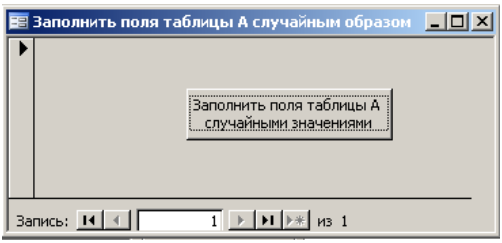



Рисунок В.1 Проста форма для ініціювання функції генерації значень у таблиці.

```
Function random_write() As Integer
 Randomize
  Dim mydb As Database, myrs As Recordset
 Dim r As Long, i As Long, s As String, rr As Integer
 Set mydb = CurrentDb
 Set myrc = CurrentDb.OpenRecordset("A")
 s = InputBox("Введите количество записей", "Случайный ввод данных")
 r = CLng(s)
For i = 1 To r
  s = ""
  For j = 1 To Int(Rnd * 50) 'До 50 символів у полі
   rr = Int(65 + (Rnd * 57))  'Символи - довільні латинські
   s = s + Chr(rr)
  Next j
' Створення нового запису. Значення поля лічильника визначається автоматично
```

```
 myrc.AddNew
 ' Додання значення текстовому полю
 myrc!a2 = s
' Додання значення числовому полю
 myrc!a3 = Int(Rnd * (10 ^ Int(Rnd * 10)))
 myrc.Update
 Next
End Function
```

Методичне видання
(українською мовою)

Кеберле Наталія Геннадіївна

# БАЗИ ДАНИХ ТА ІНФОРМАЦІЙНІ СИСТЕМИ

Методичні рекомендації до виконання курсових робіт
для студентів напряму підготовки 6.040302 – "Інформатика"

Рецензент Єрмолаєв В.А.
Відповідальний за випуск: Борю С.Ю.
Коректор Кеберле Н.Г.

Підп. до друку _____. Формат 60×90/16. Папір офсетний.
Друк різографічний. Умовн.друк.арк. 3,7.
Замовлення № ___. Наклад 20 прим.

_____
Запорізький державний університет

69600, м. Запоріжжя, МСП-41
вул. Жуковського, 66

Свідоцтво про внесення до Державного реєстру
ДК № 2952 від 30. 08. 2007 р.