**ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ОЛЕСЯ ГОНЧАРА**

Кафедра електронних обчислювальних машин

**ЗВІТ З ПРАКТИКИ**

Студента II курсу групи КІ-21-2

Спеціальності 123 – «Комп’ютерна інженерія»

Шейко Р.О

(прізвище та ініціали)

Керівник:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів

Національна шкала

Члени комісії:

Пономарьов І.В.

(підпис)

(прізвище та ініціали)

Гниленко О.Б.

(підпис)

(підпис)

(прізвище та ініціали)

(прізвище та ініціали)

Мазурик С.В.

м. Дніпро

2023

**ЗМІСТ**

[**ВСТУП** 2](#_Toc138673084)

[**ЛІТЕРАТУРНИЙ ОГЛЯД З ВИЗНАЧЕННЯ СУЧАСНИХ ЗАСОБІВ** 3](#_Toc138673085)

[**ПРАКТИЧНА ЧАСТИНА** 5](#_Toc138673086)

[**1.1.** **Умова завдання** 5](#_Toc138673087)

[**1.2.** **Збір вимог до системи** 5](#_Toc138673088)

[**1.3.** **Моделювання вимог** 6](#_Toc138673089)

[**1.4.** **Визначення об’єктів та атрибутів БД** 6](#_Toc138673090)

[**1.5.** **Перевірка того, що БД знаходиться в 1НФ, 2НФ і 3НФ** 8](#_Toc138673091)

[*1.5.1.* *Приведення таблиць до 1НФ* 8](#_Toc138673092)

[*1.5.2.* *Приведення таблиць до 2НФ* 10](#_Toc138673093)

[*1.5.3.* *Приведення таблиць до 3НФ* 12](#_Toc138673094)

[*1.5.4.* *Приведення таблиць до 4НФ* 14](#_Toc138673095)

[*1.5.5.* *Денормалізація даних* 15](#_Toc138673096)

[**1.6.** **Cтворення БД в СУБД** 16](#_Toc138673097)

[**1.7.** **Визначення зв’язків між таблицями** 19](#_Toc138673098)

[**1.8.** **Створення запитів** 20](#_Toc138673099)

[**1.9.** **Розробка інтерфейсу** 23](#_Toc138673100)

[**1.10.** **Створення звітів** 24](#_Toc138673101)

[**ВИСНОВОК** 27](#_Toc138673102)

[**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ** 28](#_Toc138673103)

# **ВСТУП**

Проектно-технологічна практика, яка є частиною навчального плану підготовки бакалаврів за спеціальністю "Комп'ютерна інженерія", має на меті надати студентам можливість поглибити та застосувати свої знання, отримані під час вивчення фундаментальних та спеціальних дисциплін, таких як "Програмування", "Системне програмування" і "Організація баз даних". Ця практика дозволяє студентам оволодіти навичками розробки баз даних та інтерфейсів до них, що сприятиме розробці прототипів інформаційних систем.

У рамках проектно-технологічної практики студентам ставиться завдання розробити інформаційну систему для будівельної компанії. Основна мета цього завдання полягає в систематизації, закріпленні та розширенні теоретичних і практичних знань, які були отримані під час навчання на попередніх курсах.

Основною проблемою, яку необхідно вирішити студенту-практиканту, є розробка імовірної бази даних та створення програмного додатку, який забезпечуватиме введення інформації до цієї бази даних, а також модуль формування звітності. Дана інформаційна система повинна враховувати потреби будівельної компанії та забезпечувати ефективне керування співробітниками, посадами, видами робіт, матеріалами, бригадами, замовниками та замовленнями.

У процесі проходження практики студент повинен вивчити стан питань з визначеної проблеми, поглибити свої знання за вищезгаданими дисциплінами та консультуватися з керівником практики для обговорення плану дій. Після уточнення календарного плану робіт, студент розпочинає вирішення поставленої задачі, проводить обробку та аналіз отриманих даних, формулює висновки та регулярно звітує про прогрес у виконанні завдання. На завершальних етапах практики студент оформляє письмовий звіт та захищає результати своєї роботи.

Результати проектно-технологічної практики допоможуть студентам успішно засвоїти дисципліни на наступних курсах навчання, а також розвинути їх уміння та навички, необхідні для вирішення професійних задач у майбутньому.

# **ЛІТЕРАТУРНИЙ ОГЛЯД З ВИЗНАЧЕННЯ СУЧАСНИХ ЗАСОБІВ**

У сучасному світі існує велика кількість різноманітних засобів, які можуть бути використані для практичного вирішення завдання розробки інформаційної системи для будівельної компанії. Зокрема, одним з найпоширеніших програмних засобів для створення та управління базами даних є Microsoft Access.

Microsoft Access є реляційною системою керування базами даних (СКБД), яка надає широкий спектр можливостей для розробки інформаційних систем. Access має зручний інтерфейс, що дозволяє легко створювати таблиці, форми, запити та звіти. Він підтримує стандартні мови запитів, такі як SQL, що дозволяє виконувати потужні запити до бази даних.

Однією з переваг Microsoft Access є його швидкість розробки. Завдяки інтуїтивно зрозумілому інтерфейсу та готовим шаблонам баз даних, створення інформаційної системи може здійснюватись швидко і ефективно. Access також надає засоби для визначення зв'язків між таблицями, налаштування прав доступу до даних і забезпечення їх безпеки.

Ще однією важливою особливістю Access є його інтеграція з іншими продуктами Microsoft, такими як Excel, Word та Outlook. Це дозволяє легко обмінюватися даними між різними програмами та використовувати їх для створення звітів, листів тощо.

Окрім Microsoft Access, існують інші альтернативні програмні засоби для розробки баз даних, такі як MySQL, Oracle, PostgreSQL та інші. Кожен з цих засобів має свої переваги та особливості, і вибір залежить від конкретних потреб та вимог проекту

У контексті практичного вирішення задачі для будівельної компанії, Microsoft Access може бути відмінним вибором. Він надає зручні інструменти для організації даних про співробітників, посади, види робіт, матеріали, бригади, замовників та замовлення. За допомогою Access можна легко створити форми для введення та редагування даних, запити для отримання необхідної інформації та звіти для звітності.

Отже, Microsoft Access є потужним програмним засобом, який може бути використаний для практичного вирішення задачі розробки інформаційної системи для будівельної компанії. Його зручний інтерфейс, швидкість розробки, можливість інтеграції з іншими продуктами Microsoft та широкий функціонал роблять його привабливим вибором для студентів-практикантів, що займаються даною проблемою.

# **ПРАКТИЧНА ЧАСТИНА**

* 1. **Умова завдання**

Тема № 19: Будівельна компанія

Таблиці:

1) Співробітники (код, ПІБ, дата народження, стать, адреса, телефон, паспорт).

2) Посади (код, найменування, оклад, обов'язки, вимоги).

3) Види робіт (код, найменування, опис, ціна).

4) Матеріали (код, найменування, упаковка, опис, ціна).

5) Бригади (код, дата створення, спеціалізація, бригадир).

6) Замовники (код, ПІБ, адреса, телефон).

7) Замовлення (код, замовник, опис, вартість, дата початку, дата закінчення).

Виконані роботи, Оплати замовлень та інші таблиці за необхідності.

Запити і подання:

1) Штат працівників компанії.

2) Виконані роботи по замовленням.

3) Використані матеріали по замовленням.

4) Незакінчені замовлення.

Інші запити та подання для покриття потреб бізнесу.

* 1. **Збір вимог до системи**

СУБД «Будівельна компанія» повинна надавати можливість користувачу:

* + - змінювати дані;
    - зберігати дані;
    - ані;
    - додавати дані;
    - видаляти дані;
    - робити звіт;
    - виконувати запити.

Також потрібно, щоб були створенні таблиці до того варіанта завдання, що було

Видано.

* 1. **Моделювання вимог**

На рис. 1 наведено моделювання вимог користувача до СУБД "Будівельна компанія" за допомогою діаграми варіантів використання (use-case diagram). Оскільки кожному користувачеві буде надано однаковий функціонал, то у нашому випадку буде лише один актор - користувач. Він може виконувати наступні дії (прецеденти): змінювати дані, зберігати дані, переглядати дані, додавати дані, видаляти дані, робити звіти та виконувати запити.

Изображение выглядит как текст, зарисовка, круг, рисунок

Автоматически созданное описание

Рис. 1. – Моделювання вимог користувача до СУБД "Будівельна компанія" за допомогою діаграми use-case

Кожен прецедент представляє окрему функцію в системі, яка складається з послідовності виконуваних дій. Поведінку не тільки системи, але й окремої функції можна змоделювати за допомогою діаграми активності (Activity diagram). Ця діаграма подібна до блок-схеми, де використовуються спеціальні символи, такі як чорний круг та "котяче око". Діаграма активності підтримує використання оператору switch (множинний вибір) та розпаралелювання потоків, що є корисним при паралельних та розподілених обчисленнях.

* 1. **Визначення об’єктів та атрибутів БД**

Для початку потрібно зрозуміти, що таке реляційна база даних та яка в неї структура. Реляційна база даних (РБД) є організованою сукупністю взаємозв'язаних таблиць, яка забезпечує зберігання та управління даними. У РБД кожна таблиця містить інформацію про об'єкти певного виду, де кожен рядок таблиці представляє дані про один об'єкт, а стовпці містять характеристики цих об'єктів - атрибути. Рядки таблиці називаються записами, а поля запису зберігають атрибути об'єкту. Первинний ключ використовується для ідентифікації записів.

Робота з реляційною базою даних відбувається за допомогою систем управління базами даних (СУБД). Основні функції СУБД включають визначення структури бази даних, обробку даних та керування даними. Структура бази даних, яка описується мовою визначення даних СУБД, є важливим проектним рішенням при створенні додатків, використовуючи базу даних.

СУБД дозволяють виконувати операції з даними, такі як додавання, видалення та оновлення записів у таблицях. Запити до бази даних формуються за допомогою мови структурованих запитів (SQL) і можуть виконувати пошук записів, що відповідають певним критеріям, або вносити зміни в таблиці.

Крім того, управління даними включає захист від несанкціонованого доступу до даних, підтримку багатокористувацького режиму роботи та забезпечення цілісності та узгодженості даних.

Розглянемо на прикладі одну з таблиць «Замовлення».

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 2. – Конструктор таблиці «Замовлення»

В першому полі з назвою «Код» ми маємо тип даних лічильник й є ключом(такого ж значення не існує). Тобто, кожен запис у цій таблиці має свій порядковий номер. По номеру запису виконується звернення до таблиць та їх елементів, тому що вони є унікальними.

В другому полі «Код\_замовника» діє підстановка, за допомогою такого коду:

SELECT [Людина].[Код], [Людина].[Прізвище] FROM Людина;

З таблиці «Людина» підтягується код елементу в тій таблиці та прізвище людини й ми отримуємо стовбчик «Код\_замовника».

В третьому полі «Опис» з типом даних короткий текст записаний опис до замовлення. Воно використовується для передачі в інші поля з кодом й описує роботу.

В четвертому полі «Вартість» з типом даних грошовий записано, скільки коштує це замовлення.

В п'ятому «Дата\_початку» та шостому «Дата\_закінчення» полях з типом даних дата та час записано однойменні дати початку/завершення робіт з замовленням.

Таким чином працює таблиця. На рис. 3 наведено те, що зберігається у таблиці.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рис. 3. – таблиця «Замовлення»

* 1. **Перевірка того, що БД знаходиться в 1НФ, 2НФ і 3НФ**
     1. *Приведення таблиць до 1НФ*

Порівняння таблиці з 1НФ та приведення її до цієї форми є важливим етапом в процесі проектування бази даних. 1НФ (перша нормальна форма) є однією з основних нормалізаційних форм, яка встановлює правила для структури таблиці, щоб уникнути повторення даних і забезпечити унікальність записів.

У таблиці, що не відповідає 1НФ, можуть бути повторювані рядки або набори значень, що приводить до неправильного організації даних і ускладнює подальшу роботу з ними. Наприклад, якщо ми маємо таблицю "Співробітників" і в ній є поле "Департамент", яке містить набір значень, що повторюються для кожного співробітника, це свідчить про те, що таблиця не відповідає 1НФ.

Для приведення таблиці до 1НФ потрібно провести процес нормалізації. Основна ідея полягає в розбитті початкової таблиці на дві або більше таблиць таким чином, щоб у кожній таблиці були унікальні записи і залежності між ними були чітко визначені. У нашому прикладі, для виправлення проблеми з полем "Департамент", ми можемо створити окрему таблицю "Департаменти" з унікальними значеннями кожного департаменту та використовувати ідентифікатор департаменту як зовнішній ключ у таблиці "Співробітників".

Процес нормалізації може також включати розбиття повторюваних груп полів на окремі таблиці, визначення первинних ключів та зовнішніх ключів для забезпечення зв'язків між таблицями, а також уточнення атрибутів і визначення залежностей між ними.

Після приведення таблиці до 1НФ отримуємо кращу організацію даних, уникнення повторення і забезпечення структурованості бази даних. Це дозволяє полегшити запити до бази даних, зменшити розмір таблиць та уникнути аномалій при внесенні, оновленні і видаленні даних.

Наприклад, в моїй базі даних «Співробітники» зв’язані з «Людиною» і це є одним із найяскравіших прикладів 1НФ. Кожний запис в цій таблиці є унікальним. Приклад схеми на рис. 4.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рис. 4 – приклад 1НФ

* + 1. *Приведення таблиць до 2НФ*

Приведення таблиці з 1НФ (першої нормальної форми) до 2НФ (другої нормальної форми) є важливим кроком при проектуванні бази даних. Давайте розглянемо процес приведення таблиці з 1НФ до 2НФ та відмінності між ними.

1. Перша нормальна форма (1НФ):

В таблиці, що знаходиться в 1НФ, кожне поле (стовпець) містить тільки атомарні значення, тобто неділений атрибут. Кожен рядок (запис) унікально ідентифікується за допомогою первинного ключа, і всі атрибути в таблиці залежать від цього первинного ключа.

2. Друга нормальна форма (2НФ):

У таблиці, що знаходиться в 2НФ, кожний неключовий атрибут (стовпець) залежить від всього первинного ключа, а не від його частини. Іншими словами, якщо таблиця містить складний первинний ключ, то кожен атрибут в таблиці повинен залежати від усіх стовпців, що входять до цього ключа.

Різниця між 1НФ і 2НФ:

- 1НФ гарантує, що атрибути в таблиці є атомарними, тоді як 2НФ зосереджена на залежності неключових атрибутів від усього первинного ключа.

- У 1НФ немає пов'язаності між неключовими атрибутами, тоді як 2НФ покликана усунути часткову залежність неключових атрибутів від складного первинного ключа.

- 1НФ стосується структури таблиці, вимагаючи, щоб кожне поле містило тільки атомарні значення, тоді як 2НФ стосується функціональної залежності атрибутів.

Щоб привести таблицю з 1НФ до 2НФ, слід виконати наступні кроки:

1. Виділити всі складні а

трибути, що залежать від частини первинного ключа.

2. Створити нову таблицю для кожного виділеного атрибута разом з відповідними атрибутами первинного ключа, від якого він залежить.

3. В оригінальній таблиці замінити виділені атрибути на зовнішні ключі, посилаючись на відповідні таблиці.

Приведення таблиці з 1НФ до 2НФ допомагає забезпечити більш ефективну організацію даних і уникнути проблем, пов'язаних з некоректною функціональною залежністю. Цей процес є частиною нормалізації баз даних, яка дозволяє покращити їх проектування та продуктивність.

Наведу приклад приведення. Зважаючи на мою відповідь вище, ось приклад переведення таблиці з 1НФ до 2НФ:

Припустимо, що у нас є таблиця з іменем "Книги" (Books) з такими стовпцями:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код книги | Назва книги | Автор | Жанр | Рік видання |
| 1 | Book A | Author 1 | Fantasy | 2005 |
| 2 | Book В | Author 2 | Thriller | 2010 |
| 3 | Book С | Author 1 | Romance | 2018 |

Ця таблиця знаходиться в 1НФ, оскільки кожен стовпець містить атомарне значення, а кожен запис унікально ідентифікується за допомогою коду книги (первинного ключа).

Але вона не задовольняє 2НФ, оскільки стовпець "Жанр" залежить від автора, а не від усього первинного ключа ("Код книги"). Тобто ми маємо часткову залежність неключового атрибута від складного первинного ключа.

Щоб перевести цю таблицю в 2НФ, ми виконаємо такі кроки:

1. Виділимо стовпець "Жанр" як неключовий атрибут, що залежить від частини первинного ключа "Автор".

Тепер ми створимо нову таблицю з назвою "Автори та жанри" (AuthorsGenres) з такими стовпцями:

|  |  |
| --- | --- |
| Автор | Жанр |
| Author 1 | Fantasy |
| Author 1 | Thriller |
| Author 1 | Romance |

2. В оригінальній таблиці "Книги" замінимо стовпець "Жанр" на зовнішній ключ, посилаючись на таблицю "Автори та жанри".

Тепер, таблиця "Книги" буде мати такий вигляд:

|  |  |  |  |
| --- | --- | --- | --- |
| Код книги | Назва книги | Автор | Рік видання |
| 1 | Book A | Author 1 | 2005 |
| 2 | Book В | Author 2 | 2010 |
| 3 | Book С | Author 1 | 2018 |

Таким чином, ми перевели таблицю з 1НФ до 2НФ, розбивши її на дві таблиці із збереженням зв'язку між ними за допомогою зовнішнього ключа. Тепер ми уникнули часткової залежності неключового атрибута від складного первинного ключа.

Наприклад, в моїй базі даних «Матеріали до замовлення» зв’язані з «Матеріалами» і це є одним із найяскравіших прикладів 2НФ. Кожний запис в цій таблиці є унікальним. Приклад схеми на рис. 5. Наприклад, а раніше забагато полей було в матеріалах тому само та умова не виконувалась.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 5 – приклад 2НФ

* + 1. *Приведення таблиць до 3НФ*

Зважаючи на мою попередню відповідь, ось текст про приведення таблиці з 2НФ до 3НФ та опис різниці між ними:

Приведення таблиці з 2НФ (другої нормальної форми) до 3НФ (третьої нормальної форми) полягає в усуненні транзитивних залежностей між неключовими атрибутами.

Розглянемо приклад таблиці "Книги" (Books), що знаходиться в 2НФ з такими стовпцями:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код книги | Назва книги | Автор | Рік видання | Жанр |
| 1 | Book A | Author 1 | 2005 | Fantasy |
| 2 | Book B | Author 2 | 2010 | Thriller |
| 3 | Book C | Author 1 | 2018 | Romance |

Ця таблиця вже задовольняє 2НФ, оскільки всі неключові атрибути пов'язані з усіма частинами первинного ключа ("Код книги", "Автор"). Однак, вона може мати транзитивну залежність між неключовими атрибутами.

У нашому прикладі стовпець "Жанр" залежить від стовпця "Автор", а не від цілого первинного ключа. Це означає, що у нас є транзитивна залежність між атрибутами: "Код книги" -> "Автор" -> "Жанр".

Для переведення таблиці в 3НФ, ми розбиваємо її на дві окремі таблиці та усуваємо транзитивну залежність. Ось кроки, які потрібно виконати:

1. Створюємо нову таблицю "Автори" (Authors) з такими стовпцями:

|  |
| --- |
| Автор |
| Author 1 |
| Author 2 |

2. У таблиці "Книги" замінюємо стовпець "Автор" на зовнішній ключ, посилаючись на таблицю "Автори".

Тепер, таблиця "Книги" буде мати такий вигляд:

|  |  |  |  |
| --- | --- | --- | --- |
| Код книги | Назва книги | Автор (зовнішній ключ) | Рік видання |
| 1 | Book A | Author 1 | 2005 |
| 2 | Book B | Author 2 | 2010 |
| 3 | Book C | Author 1 | 2018 |

3. У таблиці "Книги" залишається тільки стовпець, що є прямим ключем ("Код книги") та неключові атрибути ("Назва книги", "Рік видання").

Таким чином, ми перевели таблицю з 2НФ до 3НФ, усунувши транзитивну залежність між неключовими атрибутами. Тепер кожна таблиця містить самостійну інформацію без зайвих залежностей, що поліпшує нормалізацію даних та їх ефективне управління.

Наприклад, в моїй базі даних «Матеріали до замовлення» зв’язані з «Матеріалами» і це є одним із найяскравіших прикладів 3НФ. Кожний запис в цій таблиці є унікальним. Приклад схеми на рис. 5. Наприклад, а раніше забагато полей було в матеріалах тому саме ця умова не виконувалась.

* + 1. *Приведення таблиць до 4НФ*

Приведення таблиці з 3НФ (третьої нормальної форми) до 4НФ (четвертої нормальної форми) виникає, коли ми маємо багатозначні залежності між неключовими атрибутами. Це означає, що в таблиці існують ситуації, коли один неключовий атрибут визначає інший неключовий атрибут.

Розглянемо приклад таблиці "Студенти" (Students), яка знаходиться в 3НФ з такими стовпцями:

|  |  |  |  |
| --- | --- | --- | --- |
| Студент | Курс | Факультет | Викладач |
| John | 1 | Engineering | Professor Smith |
| Mary | 2 | Science | Professor Brown |
| Peter | 1 | Engineering | Professor Smith |

У цій таблиці ми можемо помітити багатозначну залежність між атрибутами "Факультет" та "Викладач". Тобто, одному факультету може відповідати декілька викладачів, а кожному викладачу може відповідати декілька факультетів.

Для переведення таблиці в 4НФ ми розбиваємо її на дві окремі таблиці і усуваємо багатозначну залежність. Ось кроки, які потрібно виконати:

1. Створюємо нову таблицю "Факультети" (Faculties) з такими стовпцями:

|  |
| --- |
| Факультет |
| Engineering |
| Science |

2. Створюємо ще одну таблицю "Викладачі" (Professors) з такими стовпцями:

|  |
| --- |
| Викладач |
| Professor Smith |
| Professor Brown |

3. У таблиці "Студенти" залишається тільки стовпці, що є ключами ("Студент", "Курс") та стовпці, що є неключовими атрибутами ("Факультет").

Таким чином, ми перевели таблицю з 3НФ до 4НФ, усунувши багатозначну залежність між неключовими атрибутами. Кожна таблиця містить самостійну інформацію без зайвих залежностей, що сприяє більш ефективній нормалізації та управлінню даними.

4НФ дозволяє подальше усунення аномалій залежності відносно змінюваних множин атрибутів, що є корисним для побудови дуже гнучкої та розширюваної бази даних.

Наприклад, в моїй базі даних «Матеріали до замовлення» зв’язані з «Матеріалами» і це є одним із найяскравіших прикладів 2НФ. Кожний запис в цій таблиці є унікальним. Приклад схеми на рис. 5. Наприклад, а раніше забагато полей було в матеріалах тому само та умова не виконувалась.

* + 1. *Денормалізація даних*

Денормалізація є процесом об'єднання таблиць з більш високої нормальної форми (у нашому випадку, 4НФ) у менш високу нормальну форму (у нашому випадку, 3НФ). Цей процес виконується з метою покращення продуктивності бази даних та спрощення запитів до даних.

Однією з основних причин для денормалізації з 4НФ до 3НФ є оптимізація продуктивності. Нормалізація до високих нормальних форм може призвести до багатьох зв'язків між таблицями, що вимагає складних запитів з використанням з'єднань (joins) для отримання потрібних даних. Це може призвести до погіршення продуктивності бази даних, особливо при роботі з великими обсягами даних. Денормалізація може зменшити кількість з'єднань та спростити запити, тим самим підвищуючи швидкодію системи.

Іншою причиною для денормалізації може бути покращення зручності використання даних. У високо нормалізованій базі даних існує розподіл інформації між багатьма таблицями, що може ускладнити складання запитів та розуміння структури даних. За допомогою денормалізації, можна об'єднати пов'язані дані в одну таблицю, що спрощує розуміння та використання даних.

Однак, важливо зазначити, що денормалізація також може мати певні недоліки. Вона може призвести до дублювання даних, оскільки інформація, яка раніше була розподілена між різними таблицями, тепер буде зберігатися в одній таблиці. Це може збільшити обсяг пам'яті, потрібний для зберігання даних, а також вимагатиме додаткових зусиль для підтримки цілісності даних.

Окрім того, денормалізація може знизити гнучкість бази даних. У високо нормалізованій базі даних зміни в структурі даних можна легко виконати, оскільки інформація розподілена між різними таблицями. Однак, при денормалізації, зміни в структурі таблиці можуть бути складнішими, оскільки дані зберігаються в одній таблиці.

Отже, при денормалізації з 4НФ до 3НФ важливо розглядати баланс між продуктивністю бази даних та зручністю використання даних. Цей процес може бути корисним у випадках, коли швидкодія системи має велике значення, або коли спрощення структури даних полегшує розуміння та використання бази даних.

* 1. **Cтворення БД в СУБД**

Для початкустворимо усі потрібні таблиці:

|  |  |
| --- | --- |
|  |  |
| Рис. 6а – конструктор таблиці «Бригада» | Рис. 6б – таблиця «Бригада» |

|  |  |
| --- | --- |
|  |  |
| Рис. 7а – конструктор таблиці «Види\_робіт» | Рис. 7б – таблиця «Види\_робіт» |

|  |  |
| --- | --- |
|  |  |
| Рис. 8а – конструктор таблиці «Замовлення» | Рис. 8б – таблиця «Замовлення» |

|  |  |
| --- | --- |
|  |  |
| Рис. 9а – конструктор таблиці «Людина» | Рис. 9б – таблиця «Людина» |

|  |  |
| --- | --- |
|  |  |
| Рис. 10а – конструктор таблиці «Матеріали» | Рис. 10б – таблиця «Матеріали» |

|  |  |
| --- | --- |
|  |  |
| Рис. 11а – конструктор таблиці «Матеріали до замовлення» | Рис. 11б –таблиця «Матеріали до замовлення» |

|  |  |
| --- | --- |
|  |  |
| Рис. 12а – конструктор таблиці «Посада» | Рис. 12б –таблиця «Посада» |

|  |  |
| --- | --- |
|  |  |
| Рис. 13а – конструктор таблиці «Роботи бригад» | Рис. 13б –таблиця «Роботи бригад» |

|  |  |
| --- | --- |
|  |  |
| Рис. 14а – конструктор таблиці «Роботи для замовлень» | Рис. 14б – таблиця «Роботи для замовлень» |

|  |  |
| --- | --- |
|  |  |
| Рис. 15а – конструктор таблиці «Співробітники» | Рис. 15б – таблиця «Співробітники» |

|  |  |
| --- | --- |
|  |  |
| Рис. 16а – конструктор таблиці «Трудова книжка» | Рис. 16б – таблиця «Трудова книжка» |

* 1. **Визначення зв’язків між таблицями**

В Access, зв'язки між таблицями встановлюються для побудови взаємозв'язків та забезпечення цілісності даних. Основні типи зв'язків, які можна встановити в Access, включають:

1. Один до одного (One-to-One): У цьому типі зв'язку, кожен запис в одній таблиці відповідає одному запису в іншій таблиці. Цей тип зв'язку зазвичай використовується для деталей або розширень головної таблиці.

2. Один до багатьох (One-to-Many): Це найпоширеніший тип зв'язку в Access. У ньому кожен запис в одній таблиці може мати багато відповідних записів в іншій таблиці. Наприклад, у базі даних клієнтів кожен клієнт може мати багато замовлень.

3. Багато до багатьох (Many-to-Many): В Access цей тип зв'язку реалізується за допомогою проміжних таблиць. Він використовується, коли багато записів в одній таблиці відповідає багатьом записам в іншій таблиці. Наприклад, у базі даних студентів може бути багато студентів, які вчаться на багатьох курсах, і в той же час, кожен курс може мати багато студентів.

Зв'язки в Access встановлюються за допомогою ключів. Зазвичай, головна таблиця має первинний ключ, який використовується для посилання на зовнішні ключі в інших таблицях. Зовнішні ключі вказують на пов'язані записи в інших таблицях.

Встановлення зв'язків в Access може бути виконане через візуальний редактор зв'язків або за допомогою мови структуризованих запитів (SQL). Ці зв'язки допомагають уникнути дублювання даних, забезпечують цілісність бази даних та спрощують запити та звіти, що використовують дані з різних таблиць.

Визначимо зв’язки між таблицями:

Изображение выглядит как текст, снимок экрана, диаграмма, число

Автоматически созданное описание

Рис. 17 – Схема зв’язків таблиці

* 1. **Створення запитів**

В Access запити використовуються для отримання, фільтрації, сортування та аналізу даних з однієї або декількох таблиць. Запити дозволяють вам створювати нові таблиці на основі існуючих даних, об'єднувати дані з різних джерел та виконувати різні обчислення.

В Access доступні різні типи запитів, які ви можете використовувати залежно від вашої потреби:

1. Запит на вибірку (Select Query): Цей тип запиту використовується для вибірки даних з однієї або декількох таблиць. Ви можете задати критерії фільтрації, сортування та вибрати конкретні поля, які потрібно вивести.

2. Запит на оновлення (Update Query): Цей тип запиту використовується для оновлення даних в таблиці. Ви можете вказати нові значення для певних полів або виконати обчислення на основі існуючих даних.

3. Запит на додавання (Append Query): Цей тип запиту використовується для додавання даних з однієї або декількох таблиць в іншу таблицю. Ви можете вибрати поля, які потрібно додати, та задати критерії фільтрації для вибору певних записів.

4. Запит на видалення (Delete Query): Цей тип запиту використовується для видалення записів з таблиці на основі певних критеріїв.

5. Запит на створення таблиці (Make Table Query): Цей тип запиту дозволяє створити нову таблицю на основі результатів іншого запиту. Ви можете вибрати поля, які потрібно включити в нову таблицю, та вказати критерії фільтрації.

6. Запит на об'єднання (Join Query): Цей тип запиту використовується для об'єднання даних з двох або більше таблиць на основі спільних полів. Ви можете вибрати, які поля потрібно включити в результати запиту та встановити правила з'єднання.

7. Запит на перехресну таблицю (Cross Tab Query): Цей тип запиту дозволяє створити перехресну таблицю, яка показує агреговані дані залежно від різних категорій.

Створення запитів в Access зазвичай виконується за допомогою інтерактивного режиму конструктора запитів, де ви можете вибрати необхідні таблиці, поля, критерії та інші параметри. Також ви можете використовувати SQL (Structured Query Language) для написання складніших запитів безпосередньо в редакторі SQL.

Створення ефективних запитів допомагає вам отримувати потрібні дані з бази даних та аналізувати їх зручним способом. Ви можете використовувати запити для створення звітів, форм, аналітичних панелей та інших об'єктів, що сприяють управлінню даними в Access.

Наприклад, у моїй роботі було створенно запит Використанні матеріали по замовленням. Ось так він виглядає в SQL:

SELECT Матеріали\_до\_замовлення.Код\_замовлення, Матеріали\_до\_замовлення.Код\_матеріалу, Матеріали\_до\_замовлення.Кількість, Замовлення.Дата\_закінчення

FROM Матеріали INNER JOIN (Замовлення INNER JOIN Матеріали\_до\_замовлення ON Замовлення.Код = Матеріали\_до\_замовлення.Код\_замовлення) ON Матеріали.Код = Матеріали\_до\_замовлення.Код\_матеріалу

WHERE (((Замовлення.Дата\_закінчення)<Date()));

Переглянемо в конструкторі той самий запрос:

Изображение выглядит как текст, снимок экрана, число, дисплей

Автоматически созданное описание

Рис. 18 – конструктор запросу «Використанні матеріали по замовленням»

Цей опис запиту є типовим для запитів.

* 1. **Розробка інтерфейсу**

Форми в Access є інструментом для створення користувацького інтерфейсу, який дозволяє взаємодіяти з даними в базі даних. Вони створюються для спрощення введення, редагування та відображення даних в зручній формі. Ось кілька причин, чому форми в Access є важливими:

1. Взаємодія з користувачем: Форми дозволяють користувачам взаємодіяти з базою даних шляхом введення, редагування та перегляду даних у відповідні поля. Вони надають зручний інтерфейс для роботи з даними без необхідності працювати безпосередньо з таблицями бази даних.

2. Персоналізація та контроль доступу: Форми дозволяють вам настроювати інтерфейс для кожного користувача або групи користувачів. Ви можете приховувати або блокувати деякі поля, встановлювати обов'язкові поля для заповнення, обмежувати доступ до окремих функцій або даних. Це дозволяє забезпечити безпеку та забезпечити, що користувачі мають доступ лише до необхідної інформації.

3. Валідація та форматування даних: Форми дозволяють використовувати правила валідації для перевірки правильності введених даних. Ви можете встановлювати обмеження на тип даних, відповідність шаблонам, обов'язкові значення тощо. Крім того, ви можете контролювати форматування полів, таке як дати, числа, валюти тощо, для забезпечення їхньої зрозумілості та послідовності.

4. Запити та фільтрація даних: Форми дозволяють створювати запити до бази даних безпосередньо з інтерфейсу. Ви можете використовувати фільтри для обмеження відображення даних, шукати конкретні записи за певними критеріями та виконувати різноманітні операції фільтрації та сортування.

5. Покращена візуалізація та зручний доступ до даних: Форми дозволяють візуально організувати дані для зручного перегляду. Ви можете використовувати таблиці, списки, комбіновані поля, кнопки та інші елементи для створення зручного та зрозумілого інтерфейсу. Це полегшує роботу з даними та робить їх більш доступними для користувачів.

Форми в Access є потужним інструментом для створення інтерактивного інтерфейсу для роботи з базою даних. Вони спрощують роботу з даними, забезпечують безпеку, дозволяють контролювати доступ та допомагають візуалізувати дані в зручній формі.

Мною було розробленно інтерфейс для будівельної фірми. Скріншоти моєї роботи наведені нижче.

|  |  |
| --- | --- |
| Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение  Автоматически созданное описание |  |
| Рис. 19а – Головна форма | Рис. 19б – Додавання 2 форма |

Переглянемо одну з форм додавання:

Изображение выглядит как текст, снимок экрана, число, диаграмма

Автоматически созданное описание

Рис. 20. – Форма додавання до БД

На цій формі є поля, у які записуються нове найменування, опис та ціну. Потім, після закінчення створення потрібно її зберігти. Для цього на формі існує кнопка збереження. Також працюють кнопки перегляду створених дописів БД. Видалення – це права остання кнопка. Також, найвища зправа кнопка це закриття форми.

Опис є типовим.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 21. – Форма перегляду БД

Інтерфейс працює так, як і в додаванні.

Изображение выглядит как текст, электроника, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 21. – Форма перегляду запиту до БД

Інтерфейс працює так, як і в додаванні, але відсутнє видалення, збереження та зміни у отриманних даних.

* 1. **Створення звітів**

Звіти в Access є інструментом для створення структурованої та професійно виглядаючої інформації з бази даних. Вони використовуються для представлення, аналізу та візуалізації даних з різних таблиць та запитів. Основна мета звітів - надання користувачам зрозумілої та зручної інформації для прийняття рішень, аналізу даних або презентації результатів.

Ось кілька причин, чому звіти в Access є важливими:

1. Візуалізація даних: Звіти дозволяють відображати дані в зрозумілій формі, такій як таблиці, графіки, діаграми або ілюстрації. Це допомагає легше сприймати та аналізувати великі обсяги даних.

2. Фільтрація та сортування: Звіти дозволяють застосовувати фільтри та сортування до даних, щоб сконцентруватися на конкретних аспектах або встановити потрібний порядок сортування.

3. Агрегація та обчислення: Звіти дозволяють виконувати агрегаційні функції, такі як підсумовування, розрахунок середніх значень, максимумів, мінімумів тощо. Це дозволяє здійснювати розрахунки та отримувати загальну статистику на основі даних.

4. Форматування та дизайн: Звіти дозволяють вам використовувати різноманітні інструменти форматування, такі як кольори, шрифти, рамки тощо, для створення професійного вигляду звітів. Ви можете використовувати заголовки, підзаголовки, нумерацію сторінок та інші елементи для поліпшення оформлення та зрозумілості звіту.

5. Параметризовані звіти: Access дозволяє створювати параметризовані звіти, які дозволяють користувачам вводити параметри для звітування. Це дозволяє настроювати звіти для відображення певних даних відповідно до вимог користувача.

6. Експорт та друк: Звіти можна експортувати в різні формати, такі як PDF, Excel, Word, HTML тощо. Крім того, ви можете надрукувати звіти для збереження або подальшого аналізу.

Звіти в Access допомагають вам організувати та представити дані зрозумілим та інформативним способом, що сприяє зростанню продуктивності, аналізу даних та прийняттю обґрунтованих рішень.

Наприклад, у моїй роботі було створенно звіт штату працівників компанії.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 22. – Створенний звіт працівників компанії

# **ВИСНОВОК**

В результаті виконання проектно-технологічної практики була розроблена інформаційна система для будівельної компанії, що дозволяє ефективно керувати співробітниками, посадами, видами робіт, матеріалами, бригадами, замовниками та замовленнями. Під час виконання практики студентом-практикантом були проведені дослідження та здійснена розробка бази даних, включаючи визначення об'єктів та атрибутів, перевірку нормалізації та видалення багатозначних залежностей. Також було розроблено програмний додаток, який забезпечує введення інформації до бази даних і формування звітів на основі збережених даних.

У ході практики студент поглибив свої знання з програмування, системного програмування та організації баз даних. Він оволодів навичками розробки баз даних та інтерфейсів до них, а також навчився моделювати функціональні вимоги за допомогою діаграм use-case та створювати запити, підстановки та звіти на основі даних з бази.

Було розроблено додаток «Будівельна фірма», що дозволяв вести облік усіх операцій, людей, матеріалу, бригад та інших об'єктів за для потреб бізнесу.

Отже, результати проектно-технологічної практики свідчать про успішне виконання поставлених завдань і розвиток професійних навичок у студента. Ця практика дала можливість застосувати отримані знання в реальному проекті, а також підготувати студента до подальшого вивчення спеціалізованих дисциплін та вирішення професійних задач у майбутньому.

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

**1.** Язык программирования [Електронний ресурс]. Режим до-ступу: https://ru.wikipedia.org/wiki/Язык\_программирования — Загол. з екрана.

**2.** Литвинов О. А., Герасимов В. В., Карпенко Н. В. Об’єктноорієнтована розробка інформаційних систем. Монографія. Д.: Ліра, 2018. 448 с.

**3.** CRVS Digitisation Guidebook. Define System Requirements. URL: <http://www.crvs-dgb.org/en/activities/analysis-anddesign/8-define-system-requirements/>

**4.** SQL в Access: основні поняття, глосарій і синтаксис. URL: https://support.microsoft.com/uk-ua/office/sql-в-access-основніпоняття-глосарій-і-синтаксис-444d0303-cde1-424e-9a74- e8dc3e460671 (дата звернення 21.05.2022).

**5.** Access SQL: речення FROM. URL: https://support.microsoft.com/uk-ua/office/access-sql-речення-from-2a7d031c-6912- 4e8c-a644-590940ddaed1 (дата звернення 21.05.2022).

**6.** Access SQL: речення WHERE. URL: https://support.microsoft.com/uk-ua/office/access-sql-речення-where-753bbc13- debc-4b28-b527-42eb7885c862 (дата звернення 21.05.2022).

**7.** Агрегатні функції SQL. URL: https://support.microsoft.com/uk-ua/office/агрегатні-функції-sql-a810ee71-c1cd43d7-9c55-8005f1893be5 (дата звернення 21.05.2022).

**8.** About the Unified Modeling Language Specification Version 2.5.1. URL: https://www.omg.org/spec/UML/ (дата звернення 21.05.2022).

**9.** Об’єднання таблиць і запитів. URL: https://support.microsoft.com/uk-ua/office/об-єднання-таблицьі-запитів-3f5838bd-24a0-4832-9bc1-07061a1478f6 (дата звернення 21.05.2022).

**10**. Герасимов В. В., Карпенко Н. В., Литвинов О. А, Навчально-методичний посібник до проектно-технологічної практики. Монографія. Д.: Ліра, 2022. — 88 с.