##### Практична робота №6. Характеристики зв’язків між сутностями та застосування під запитів.

**Мета:** Набуття практичних навичок зв’язування таблиць за допомогою SQL-запитів, реалізації зв’язків один до багатьох та багато до багатьох.

***Методичні рекомендації***

*При вивченні теми слід звернути увагу на створення та збереження структури бази даних(БД), редагування структури, визначення типів полів, зв’язування таблиць у БД.*

1. Повторити лекції №7, 8, теоретичну частину ЛР № 4,5. Ознайомитися з теоретичною частиною цієї ПР.
2. Для своєї БД виконати описаний в методичних вказівках сценарій.
3. Створити таблицю з переліком предметів Subject (Предмети), та таблицю Grades (Оцінки) для зв’язку предметів та оцінок, яка містить оцінки студентів. Таблиця Grades містить поля: ідентифікатор рядка (Id\_Gr), який задається через автоінкремент, код студента (CodeStud), код предмета (Sub\_code), дата отримання оцінки (Gr\_date), яка задається текстовим полем, номер заняття в цей день (Gr\_lesnum), оцінка (grade).
4. Заповнити таблиці Subject та Grades, використовуючи команди INSERT.
5. Додати поле Gr\_num – номер групи в таблицю Student, використовуючи команду UPDATE.
6. Заповнити таблицю Grades для зв’язування таблиць Student та Grades,
7. Переглянути таблиці Student, Subject та Grades.
8. Розробити сценарій, аналогічний наданому в методичних вказівках, та виконати його для своєї БД, включити до нього всі SQL-запити цього сценарію, включити сценарій до звіту до ПР.
9. Для своєї БД виконати описаний в методичних вказівках сценарій. Виконати цей сценарій Надати SQL-запити, зробити скріншот отриманих результатів, який додати до SQL-запитів включити до звіту по ЛР.
10. Сформувати звіт з наданням виконаних команд та відповідними поясненнями.
11. Результати надсилати на електронну адресу викладача [t.i.lumpova@gmail.com](mailto:t.i.lumpova@gmail.com)

Файл повинен мати назву в такому форматі:

**DB<Номер групи><Номер лекції / практичної / лабораторної [літера позначення типу роботи L – лекція, P – практична, R – лабораторна]<Прізвище англійською>**. Наприклад, **DB3101R**buts.doc.

Не копіюйте фрагментів з різних інформаційних джерел, подумайте і викладіть свою точку зору. При наявності робіт -"близнюків" відповідь буде зараховуватися першому за часом надсилання.

**Строк виконання цієї роботи ІПЗ-31 - 26.10.2022**

***Контрольні запитання***

1. Що Ви розумієте під схемою бази даних?

2. Що таке підсхема бази даних?

3. Перелікувати основні типи зв'язків між елементами даних.

4. Які види зв'язків між сутностями Вам відомі?

5. Визначте основні типи асоціацій.

6. В яких випадках доцільно застосовувати зв’язок 1:1? Наведіть приклад.

**МЕТОДИЧНІ ВКАЗІВКИ**

Приклад сценарію виконання зв’язування таблиць за допомогою SQL-запитів.

1. Створити таблицю Subject – список предметів.

##### CREATE TABLE "Subject" ( "Sub\_code" INTEGER NOT NULL, "Sub\_name" TEXT, PRIMARY KEY("Sub\_code"))

2. Вести дані до таблиці

INSERT INTO Subject VALUES (1, "Математика");

INSERT INTO Subject VALUES (2, "Інформатика");

INSERT INTO Subject VALUES (3, " Українська мова ");

INSERT INTO Subject VALUES (4, " Географія ");

INSERT INTO Subject VALUES (5, " Фізична культур ");

INSERT INTO Subject VALUES (6, " Іноземна мова ");

Переглянути результати за допомогою запиту: select \* from Subject

##### 3. Створити таблицю з оцінками, через яку зв’язуються таблиці Grades, Subject та Student:

##### CREATE TABLE "Grades" (

##### "Id\_Gr" INTEGER NOT NULL UNIQUE,

##### "CodeStud" INTEGER,

##### "Sub\_code" INTEGER,

##### "Gr\_date" TEXT,

##### "Gr\_lesnum" INTEGER,

##### "grade" INTEGER DEFAULT 0,

##### PRIMARY KEY("Id\_Gr" AUTOINCREMENT),

##### FOREIGN KEY("Sub\_code") REFERENCES "Subject"("Sub\_code"),

##### FOREIGN KEY("CodeStud") REFERENCES "Student"("CodeStud")

##### );

##### 4. Заповнити таблицю Grades за наданим нижче зразком, який розроблявся для СУБД SQLite. З погляду на це дата задається як текстове поле.

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (1, 1, "10/10/2022",1,1);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (1, 2, "10/10/2022",1);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (1, 3, "10/10/2022",1,1);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (1, 4, "10/10/2022",2);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (2, 1, "10/10/2022",1,1);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (2, 2, "10/10/2022",1,3);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (2, 3, "10/10/2022",1,2);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum) VALUES (2, 4, "10/10/2022",1);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (3, 1, "10/10/2022",1,1);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum) VALUES (3, 2, "10/10/2022",1);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (3, 3, "10/10/2022",1,1);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum) VALUES (3, 4, "10/10/2022",2);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum) VALUES (4, 1, "10/10/2022",1);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (4, 2, "10/10/2022",1,3);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (4, 3, "10/10/2022",1,2);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum) VALUES (4, 4, "10/10/2022",4);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (1, 1, "11/10/2022",1,1);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum) VALUES (1, 2, "11/10/2022",1);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (1, 3, "11/10/2022",1,1);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum) VALUES (1, 4, "11/10/2022",2);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (2, 1, "11/10/2022",1,1);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (2, 2, "11/10/2022",1,3);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (2, 3, "11/10/2022",1,2);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum) VALUES (2, 4, "11/10/2022",1);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (3, 1, "11/10/2022",1,1);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum) VALUES (3, 2, "11/10/2022",1);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (3, 3, "11/10/2022",1,1);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum) VALUES (3, 4, "11/10/2022",2);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum) VALUES (4, 1, "11/10/2022",1);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (4, 2, "11/10/2022",1,3);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (4, 3, "11/10/2022",1,2);

INSERT INTO Grades (CodeStud, Sub\_code, Gr\_date, Gr\_lesnum, grade) VALUES (4, 4, "10/10/2022",4);

**Додавання рядка за допомогою назв стовпчиків**

Формат оператору

INSERT INTO *table*

(*column1*, *column2*, …, *columnN*)

VALUES(*value1*, *value2*, …, *valueN*);

*table* – це назва таблиці, в яку додається рядок;

*column1*, *column2*, …,*columnN* – список назв стовпчиків в *table*;

*value1*, *value2*, …, *valueN* – список літерних позначок або виразів, які задають значення для вказаних стовпчиків в новому рядку. Кількість значень повинно відповідати кількості стовпчиків в списку, а значення повинні бути вказані в тій же послідовності, що і назви стовпчиків.

СУБД вставляє кожне значення в стовпчик, використовуючи відповідні значення в списку. Значення *value1* додається в стовпчик *column1* нового рядка, значення *value2* – в стовпчик *column2* и т.д. Пропущеному стовпчику присвоюється значення по замовчуванню абр NULL.

*Класи для збереження даних в SQLite.* Кожне значення, що зберігається у БД SQLite має один із наступних класів.

– NULL – це значення NULL.

– INTEGER – значення є цілим, зберігається в 1, 2, 3, 4, 6 або 8 байтів залежно від величини значення.

– REAL – значення числа з плаваючою комою, збережені як IEEE 8-байтове число з рухомою точкою.

– TEXT – це текстовий рядок, збережений із використанням кодової сторінки БД (UTF8, UTF-16BE або UTF-16LE).

– BLOB значення blob, зберігаються у такому вигляді, як були введені.

SQLite клас зберігання даних є трохи більш загальним, ніж тип даних. Клас зберігання, ціле число, наприклад, включає 6 різних цілих типів даних різної довжини.

5. Додати поле номеру групи Gr\_num до таблиці Student

ALTER TABLE "Student" (

"Gr\_num" INTEGER),

FOREIGN KEY("Gr\_num") REFERENCES "groups"("gr\_id")

);

6. Заповнити поле номеру групи Gr\_num таблиці Student

UPDATE Student SET Gr\_num=10 WHERE CodeStud<3;

UPDATE Student SET Gr\_num=130 WHERE CodeStud>2;

**Характеристика зв'язків**

Структура даних може бути описана формально. Опис глобальної логічної структури бази даних називається схемою. Схема визначає всі типи елементів даних, які зберігаються в базі даних, а також усі зв'язки між ними. Схема бази даних, як правило, дуже складна. Конкретний користувач або прикладний програміст не повинен знати про схему в цілому. Така необізнаність часто навіть необхідна з точки зору безпеки даних. Програміст або користувач повинен бути інформований тільки про множину даних і зв'язків, які орієнтовані на його конкретну область.

Частина схеми отримала назву підсхеми. По суті, підсхема - це деяка організація файлів прикладного програміста. В функції СКБД входить побудова відповідних підсхем із загальної схеми і передача даних користувачам і системним програмістам. При цьому схема даних повинна бути спроектована таким чином, щоб із неї могли бути побудовані всі підсхеми по запитам користувачів або прикладних програм. Ні схема, ні підсхема не визначають методів фізичного зберігання даних.

Схеми і підсхеми представляють в вигляді діаграм, на яких зображують типи елементів даних і зв'язки між ними. Розрізняють чотири види зв'язків:

1) необов'язковий зв'язок: існування об'єктів не залежить від зв'язку;

2) можливий зв'язок: існування одного з об'єктів залежить від зв'язку;

3) умовний зв'язок: частковий вид можливого зв'язку, коли задається умова існування (наприклад, зв'язок між об'єктами СТУДЕНТ, СТИПЕНДІЯ можлива при умові відповідної успішності);

4) обов'язковий зв'язок: існування обох об'єктів залежить від зв'язку.

Односторонні зв'язки між парами елементів називаються асоціаціями, а двохсторонні - відображеннями.

Між двома сутностями А й В можливі чотири види зв'язків.

Зв’язки на фізичному рівні.

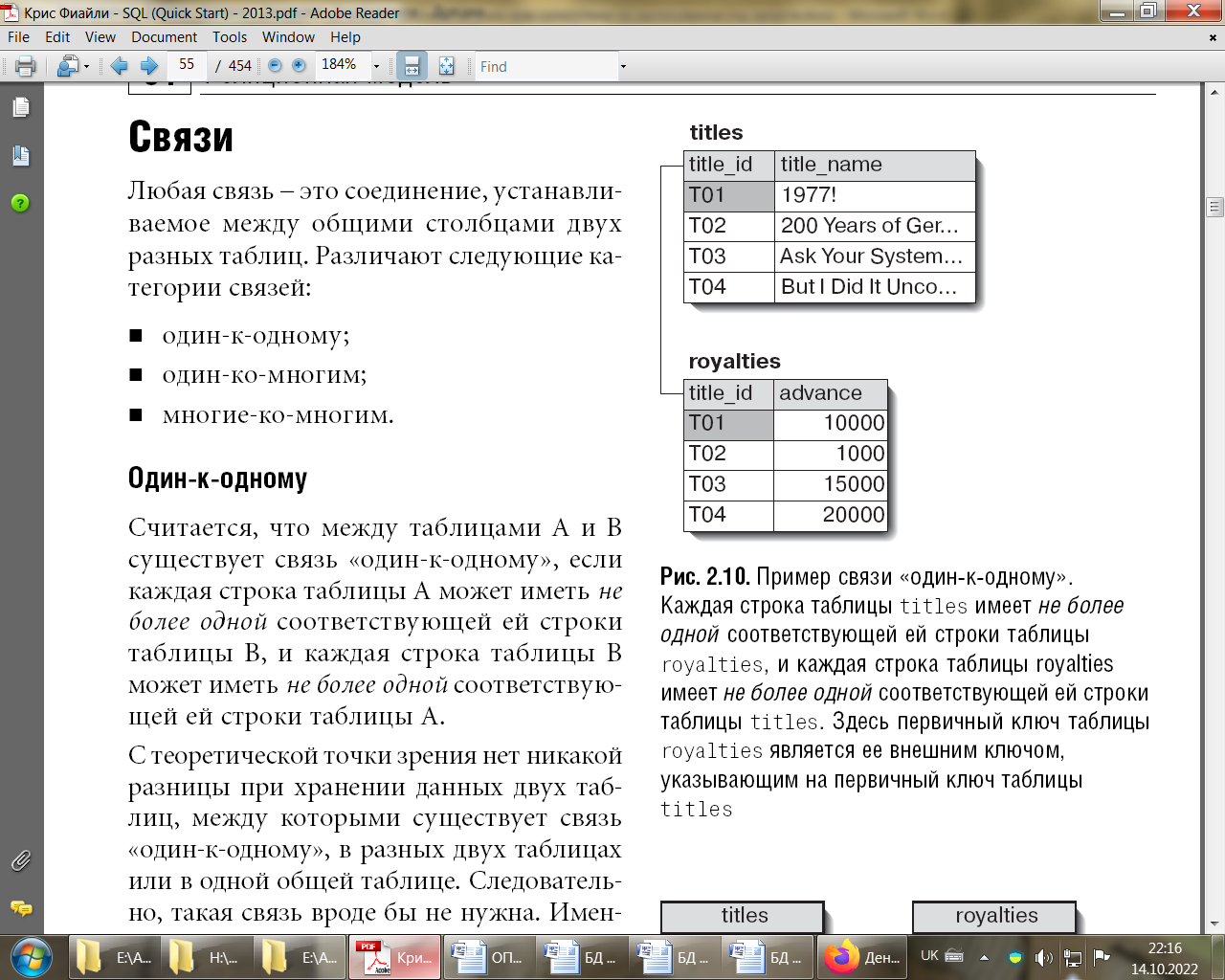
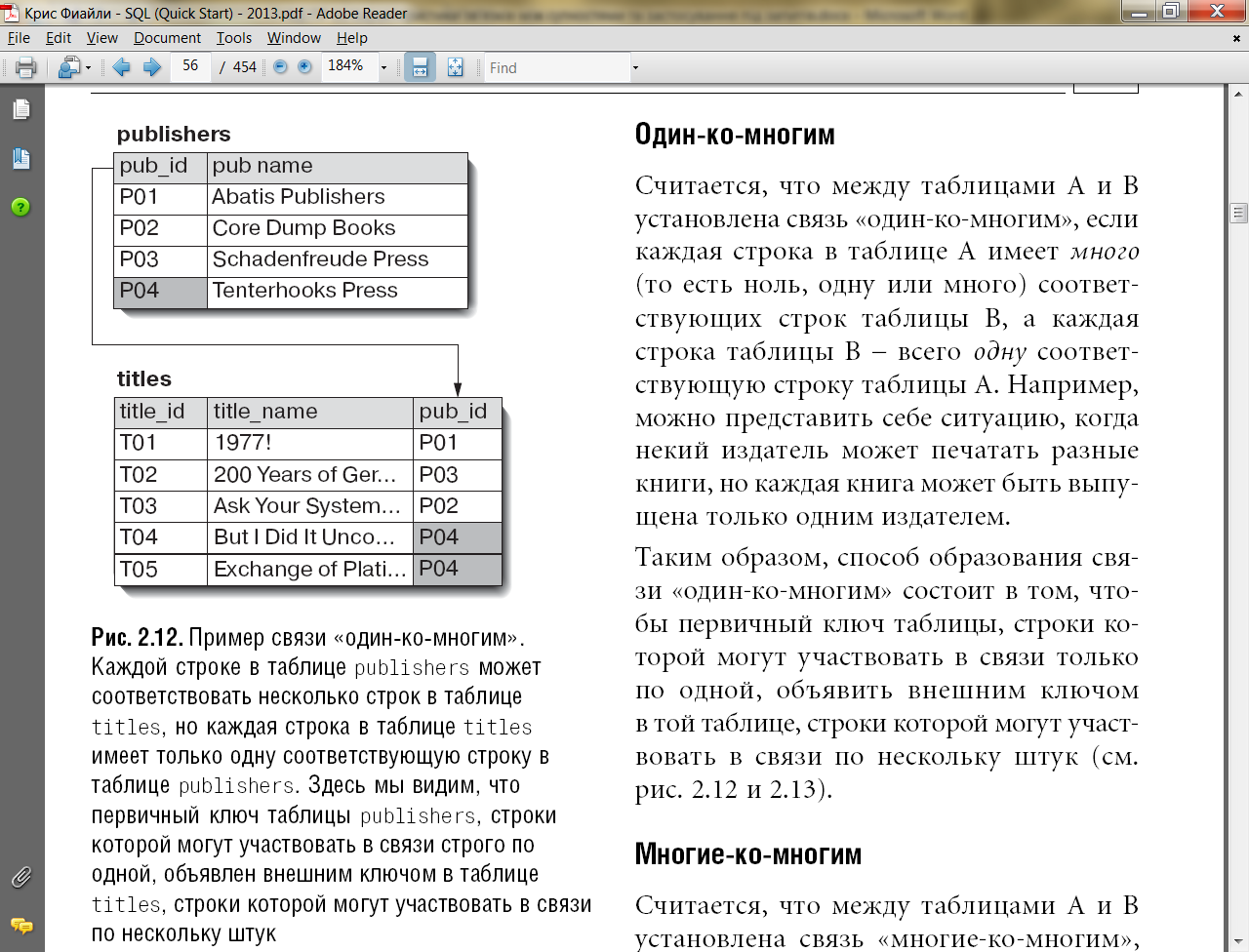
**Перший тип** - зв'язок ОДИН-ДО-ОДНОГО (1:1): за допомогою такого відображення подають такий тип зв'язку, коли в кожний момент часу кожний екземпляр елемента, від якого направлений зв'язок, ідентифікує один і тільки один екземпляр елемента, до якого направлений зв'язок, при цьому ця ідентифікація є унікальною в обох напрямках. Вважається, що між таблицями A и B існує зв'язок ОДИН-ДО-ОДНОГО, якщо кожен рядок таблиці A може мати *не більше одного* відповідного рядка таблиці B, і кожен рядок таблиці B може мати *не більше одного* відповідного рядка таблиці A. Якщо відомо значення А , то однозначно визначається і значення В. І навпаки.

Рисунок 1. Зв’язок «один-до-одного

С теоретичної точки зору нема різниці при збереженні даних двох таблиць, між якими існує зв’язок «один-до-одного», в різних двох таблицях або в одній спільній таблиці. Цей вид зв’язку застосовують з практичних міркувань або з питань безпеки, коли потрібно відділити та захистити конфіденційну інформацію, або з міркувань швидкодії для розбиття громіздких монолітних таблиць. Таким чином, спосіб встановлення зв’язку «один-до-одного» полягає в тому, щоб первинний ключ однієї таблиці оголосити зовнішнім ключем в іншій таблиці. Приклад відображення 1:1 приведений на рис.1.

**Другий тип зв’язку** - зв'язок ОДИН-ДО-БАГАТЬОХ (1:Б): якщо екземпляр елемента даних, від якого направлений зв'язок, ідентифікує деяке число екземплярів елементів даних, до яких направлений зв'язок, причому ідентифікація в даному напрямку не обов'язково є унікальною, то таке відображення називається ОДИН-ДО-БАГАТЬОХ (1:Б). Прикладом такого відношення може бути НОМЕР ВІДДІЛУ - ТАБЕЛЬНІ НОМЕРИ ПРАЦЮЮЧИХ. У відділі працює багато службовців, але кожний працюючий відноситься тільки до одного відділу. На рис..2 надано приклад за відомим нам прикладом.

Між таблицями A и B встановлений зв’язок «один-до-багатьох», якщо кожний рядок в таблиці A має *багато* (тобто нуль, одну або багато) відповідних строк таблиць B, а кожен рядок таблиці B – лише *один* відповідний рядок таблиці A. Спосіб створення зв’язку «один-до-багатьох» полягає в тому, щоб первинний ключ таблиці, рядки якої можуть приймає участь в зв’язку тільки по одному, оголосити зовнішнім ключем в той таблиці, рядки якої можуть приймати участь в зв’язку по декілька штук.

Вирази об’єднання (join) застосовуються для виконання команд SQL на декількох таблицях. Якщо додати повторювані групи до відповідної таблиці, можна побудувати зв’язок «один-до-багатьох» без третьої додаткової таблиці, але такий підхід був би порушенням першої нормальної форми. Додаткову таблицю іноді називають *асоціативною таблицею*, або *таблицею перехрещення*. Зв’язок «один-до-багатьох» ще називають зв’язок *предок*-*потомок*.

Рисунок 2. Зв’язок "один-до-багатьох"

Оскільки між двома сутностями можливі зв'язки в обох направленнях, то існує ще два типи зв'язків **БАГАТО-ДО-ОДНОГО (Б:1)** і **БАГАТО-ДО-БАГАТЬОХ (Б:Б).** Відображення Б:1 є аналогічним відображенням 1:Б. Якщо екземпляр елемента даних, від якого направлений зв'язок, ідентифікує деяке число екземплярів елементів даних, до яких направлений зв'язок, і навпаки, тобто ідентифікація не є унікальною в обох напрямках, то таке відображення називається БАГАТО-ДО-БАГАТЬОХ (Б:Б).

##### Інші приклади зв’язку БАГАТО-ДО-БАГАТЬОХ

##### 

##### Інший приклад, пацієнт, маючи одного лікаря, що лікує, може мати також декілька лікарів-консультантів; лікар може бути лікарем, що лікує для декількох пацієнтів та може одночасно консультувати декількох других пацієнтів.

##### Приклад зв’язку типу “багато до багатьох” – M:N

Нехай задано два типа сутностей: Студент та Дисципліна. Якщо студент (тип сутності Студент) вивчає деяку дисципліну (тип сутності Дисципліна), то на ER-моделі зв’язок може бути відображений так як показано на рисунку 3.

[](https://www.bestprog.net/wp-content/uploads/2019/01/03u-1.jpg)

Рисунок 3. Відображення зв’язку типу “багато до багатьох”

На рисунку 3 символами M та N позначено тип зв’язку “багато до багатьох”. Цей тип зв’язку вибрано, тому що один студент може вивчати декілька (багато) дисциплін, і, навпаки, одну дисципліну може вивчати декілька студентів.

Зв’язок “багато до багатьох” складно реалізувати програмно, тому що реляційні бази даних підтримують зв’язок “один до багатьох”. Щоб вирішити цю проблему, розробник бази даних створює штучний тип сутності, який виконує функції комутатора між двома основними сутностями.

**Приклад.** Для двох типів сутностей Студент та Дисципліна можна реалізувати штучний тип сутності, як показано на рисунку 4.

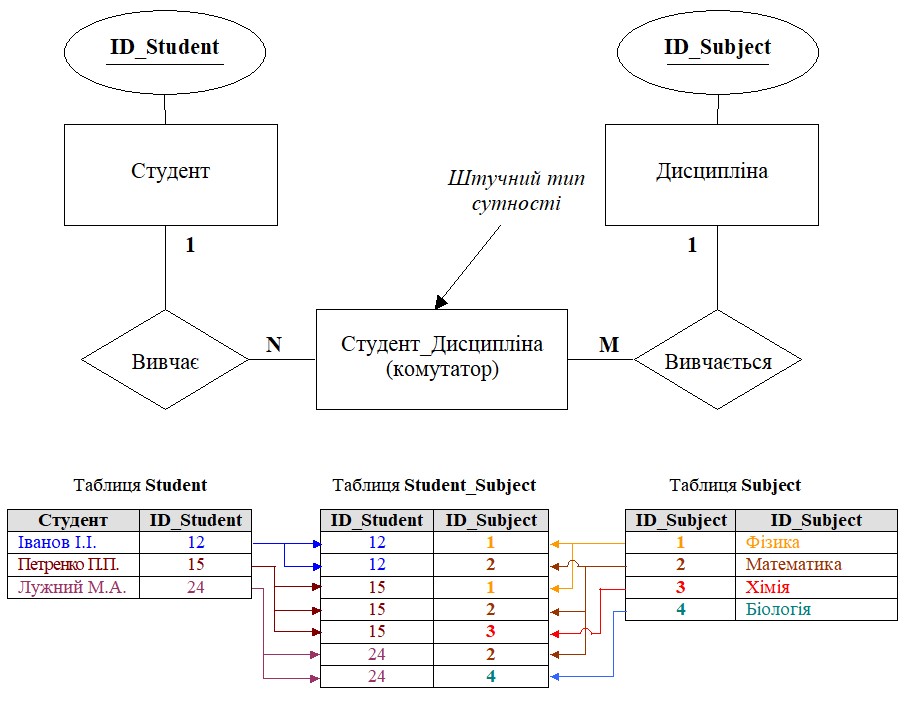
[](https://www.bestprog.net/wp-content/uploads/2019/01/04u.jpg)

Рис. 4. Реалізація зв’язку “багато до багатьох” або M:N

На вищенаведеному рисунку сутність-комутатор містить два атрибути, що є зовнішніми ключами. Один атрибут зв’язаний з підтипом сутності Студент. Другий атрибут зв’язаний з підтипом сутності Дисципліна. Також відображено наближену реалізацію відповідних реляційних таблиць. Таблиця Subject\_Student є додатково створеною таблицею, яка відображає штучний тип сутності, який виконує функції комутатора.

Для забезпечення кращої наочності вищенаведеної схеми, часто реалізується спрощений варіант, який зображено на рисунку 5. У цьому випадку штучний тип сутності зображують у вигляді ромбу, вписаного в прямокутник.

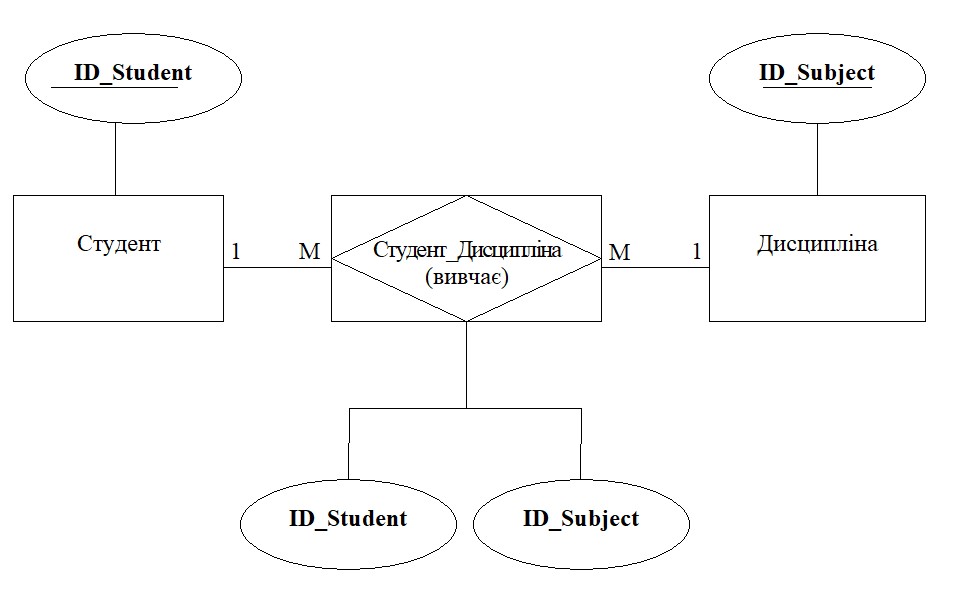
[](https://www.bestprog.net/wp-content/uploads/2019/01/05u.jpg)

Рис. 5. Реалізація спрощеного варіанту штучного типу сутності на ER-діаграмі