МІНІСТЕРСТВО ОСВІТІ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Методичні вказівки до виконання лабораторної роботи за темою «Вивчення основ роботи із засобами контролю цілісності даних засобами СУБД Microsoft SQL Server»

для студентів, що навчаються за спеціальностями
«121 Інженерія програмного забезпечення»
«122 Комп’ютерні науки»
«126 Інформаційні системи та технології»

Затверджено
редакційно-видавничою
радою університету,
протокол № 2 від 16.06.2023 р.

Харків
НТУ ХПІ
2023

Методичні вказівки до виконання лабораторної роботи за темою «Вивчення основ роботи із засобами контролю цілісності даних засобами СУБД Microsoft SQL Server» для студентів, що навчаються за спеціальностями 121 «Інженерія програмного забезпечення», 122 «Комп’ютерні науки», 126 «Інформаційні системи та технології» / Уклад. Орловський Д.Л., Копп А.М. – Харків: НТУ «ХПІ», 2023. – 40 с.

Укладачі Д.Л. Орловський,
А.М. Копп

Рецензент Гринченко М.А.

Кафедра програмної інженерії та інтелектуальних технологій управління

ЗМІСТ

ВСТУП

Сучасні інформаційні системи ґрунтуються на використанні баз даних, в яких накопичується різна інформація. Тому зараз розробляються і значно поширюються методи і засоби роботи з базами даних з метою підвищення ефективності роботи людини у різних галузях діяльності. Ці засоби та методи пов’язані з узагальненням і різними додатковими способами обробки даних. Основні ідеї сучасної інформаційної технології базуються на концепції, згідно з якою дані повинні бути організовані в бази даних з метою адекватного відображення реального світу, що змінюється, і задоволення інформаційних потреб користувачів. Ці бази даних створюються і функціонують під управлінням спеціальних програмних комплексів, які називають системами управління базами даних (СУБД).

Методичні вказівки до лабораторної роботи за темою «Вивчення основ роботи із засобами контролю цілісності даних засобами СУБД Microsoft SQL Server» призначені для студентів, що навчаються за спеціальностями «121 Інженерія програмного забезпечення», «122 Комп’ютерні науки», «126 Інформаційні системи та технології».

Виконання лабораторної роботи повинно забезпечити закріплення теоретичних знань і практичних навичок, отриманих при вивченні лекційної частини дисциплін, пов’язаних із проектуванням, розробкою та застосуванням баз даних.

В методичних вказівках розглянуті основні питання, пов’язані з теоретичним обґрунтуванням та безпосереднім виконанням лабораторної роботи.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Основні поняття цілісності даних

Виконання операторів модифікації даних (INSERT, DELETE, UPDATE) в таблицях бази даних може привести до порушення узгодженості даних і їх коректності, тобто до втрати їх достовірності і несуперечності.

Щоб інформація, що зберігається у базі даних, була однозначною і несуперечливою, в реляційній моделі встановлюються деякі обмежувальні умови - правила, що визначають можливі значення даних і таких, що забезпечують логічну основу для підтримки коректних значень. Обмеження цілісності дозволяють звести до мінімуму помилки, що виникають при оновленні і обробці даних.

У базі даних, побудованій на реляційній моделі, задається ряд правил цілісності, які, по суті, є обмеженнями для усіх допустимих станів бази даних і гарантують коректність даних.

Основні типи обмежень цілісності даних:

1. Обов'язкові дані.
2. Обмеження для доменів та атрибутів.
3. Цілісність сутностей.
4. Посилальна цілісність.
5. Вимоги замовника.

Обов'язкові дані передбачають, що деякі атрибути завжди повинні містити одно з допустимих значень, тобто, ці атрибути не можуть мати порожнього значення.

Обмеження для доменів та атрибутів передбачають, що кожен атрибут належить до певного домену, який визначає множину допустимих значень атрибуту.

Цілісність сутностей передбачає виконання низки умов, зокрема:

а) у кожній сутності повинен бути визначений первинний ключ;

б) повинна існувати можливість гарантувати унікальність значень і для будь-яких альтернативних (тобто, UNIQUE) ключів;

в) повинна існувати можливість визначення атрибутів, що є обов'язковими для заповнення (NOT NULL);

г) усі сутності в реляційній базі даних повинні бути зв’язаними. Видаленню (або зміні первинного ключа) не підлягають сутності, якщо існують об'єкти, що посилаються на них.

Посилальна цілісність передбачає існування набору правил, що забезпечують відповідність ключових значень у сутностях, зв'язаних за допомогою первинних та зовнішніх ключів.

Вказане обмеження цілісності торкається зовнішніх ключів. Зовнішній ключ – це поле (чи множина полів) однієї таблиці, що є первинним ключем іншої (чи тій же самій) таблиці. Зовнішні ключі використовуються для встановлення логічних зв'язків між таблицями. Зв'язок встановлюється шляхом привласнення значень зовнішнього ключа однієї таблиці значенням ключа інший.

Між двома або більше таблицями бази даних можуть існувати стосунки підлеглості, які визначають, що для кожного запису головної таблиці (званою ще батьківською) може існувати одна або декілька записів в підпорядкованій таблиці (званою так же дочірньої).

Існує три різновиди зв'язку між таблицями бази даних :

- «один-до-багатьох»;
- «один-до-одного»;
- «багато-до-багатьох».

Відношення «один-до-багатьох» має місце, коли одному запису батьківської таблиці може відповідати декілька записів дочірньої. Зв'язок «один-до-багатьох» іноді називають зв'язком «багато-до-одного». І у тому, і в іншому випадку суть зв'язку між таблицями залишається незмінною.

Зв'язок «один-до-багатьох» найбільш поширений для реляційних баз даних. Вона дозволяє моделювати також ієрархічні структури даних.

Відношення «один-до-одного» має місце, коли одному запису у батьківській таблиці відповідає один запис в дочірній. Це відношення зустрічається набагато рідше, ніж відношення «один-до-багатьох». Його використовують, якщо не хочуть, щоб таблиця у базі даних «розпухала» від другорядної інформації. Використання зв'язку «один-до-одного» призводить до того, що для читання пов'язаної інформації в декількох таблицях доводиться робити декілька операцій читання замість однієї, коли дані зберігаються в одній таблиці.

Відношення «багато-до-багатьох» має місце в наступних випадках:

- одному запису у батьківській таблиці відповідає більше за один запис в дочірній таблиці;
- одному запису в дочірній таблиці відповідає більше за один запис у батьківській таблиці.

Вважається, що всякий зв'язок «багато-до-багатьох» може бути замінений на зв'язок «один-до-багатьох» (один або декілька).

Часто зв'язок між таблицями встановлюється по первинному ключу, тобто значення первинного ключа однієї таблиці привласнюються значенню зовнішнього ключа іншої. Поля зовнішнього ключа не зобов'язані мати тих же імен, що і імена ключів, яким вони відповідають. Зовнішній ключ може посилатися на свою власну таблицю – у такому разі зовнішній ключ називається рекурсивним.

Посилальна цілісність визначає: якщо в таблиці існує зовнішній ключ, то його значення повинне або відповідати значенню первинного ключа деякого запису у базовій таблиці, або задаватися визначником NULL.

Існує декілька важливих моментів, пов'язаних із зовнішніми ключами. По-перше, слід проаналізувати, чи допустиме використання в зовнішніх ключах порожніх значень. У загальному випадку, якщо участь дочірньої таблиці в зв'язку є обов'язковою, то рекомендується забороняти застосування порожніх значень у відповідному зовнішньому ключі. В той же час, якщо має місце часткова участь дочірньої таблиці в зв'язку, то

приміщення порожніх значень в поле зовнішнього ключа має бути дозволене.

Наступна проблема пов'язана з організацією підтримки посилальної цілісності при виконанні операцій модифікації даних у базі. Тут можливі наступні ситуації:

1. Вставка нового рядка в дочірню таблицю. Для забезпечення посилальної цілісності необхідно переконатися, що значення зовнішнього ключа нового рядка дочірньої таблиці рівно порожньому значенню або деякому конкретному значенню, присутньому в поле первісного ключа одного з рядків батьківської таблиці.

2. Видалення рядка з дочірньої таблиці. Ніяких порушень посилальної цілісності не відбувається.

3. Оновлення зовнішнього ключа в рядку дочірньої таблиці. Цей випадок подібний до описаної вище першої ситуації. Для збереження цілісності посилання необхідно переконатися, що значення зовнішнього ключа в оновленому рядку дочірньої таблиці рівно порожньому значенню або деякому конкретному значенню, присутньому в полі первинного ключа одного з рядків батьківської таблиці.

4. Вставка рядка у батьківську таблицю. Така вставка не може викликати порушення посилальної цілісності. Доданий рядок просто стає батьківським об'єктом, що не має дочірніх об'єктів.

5. Видалення рядка з батьківської таблиці. Посилальна цілісність виявиться порушеною, якщо в дочірній таблиці існуватимуть рядки, що посилаються на видалений рядок батьківської таблиці. В цьому випадку може використовуватися одна з наступних стратегій :

NO ACTION. Видалення рядка з батьківської таблиці забороняється, якщо в дочірній таблиці існує хоч би один рядок, що посилається на неї.

CASCADE. При видаленні рядка з батьківської таблиці автоматично видаляються усі рядки дочірньої таблиці, що посилаються на неї. Якщо будь-який з рядків дочірньої таблиці, що видаляються, виступає

батьківською стороною в якому-небудь іншому зв'язку, то операція видалення застосовується до усіх рядків дочірньої таблиці цього зв'язку і так далі. Іншими словами, видалення рядка батьківської таблиці автоматично поширюється на будь-які дочірні таблиці.

SET NULL. При видаленні рядка з батьківської таблиці в усіх рядках дочірнього відношення, що посилаються на неї, в поле зовнішнього ключа, що відповідає первинному ключу видаленого рядка, записується порожнє значення. Отже, видалення рядків з батьківської таблиці викличе занесення порожнього значення у відповідне поле дочірньої таблиці. Ця стратегія може використовуватися, тільки коли в полі зовнішнього ключа дочірньої таблиці дозволяється поміщати порожні значення.

SET DEFAULT. При видаленні рядка з батьківської таблиці в поле зовнішнього ключа усіх рядків дочірньої таблиці, що посилаються на неї, автоматично поміщається значення, вказане для цього поля як значення за умовчанням. Таким чином, видалення рядка з батьківської таблиці викликає приміщення значення, що набуває за умовчанням, в поле зовнішнього ключа усіх рядків дочірньої таблиці, що посилаються на видалений рядок. Ця стратегія застосовна лише в тих випадках, коли полю зовнішнього ключа дочірньої таблиці призначено деяке значення, що приймається за умовчанням.

NO CHECK. При видаленні рядка з батьківської таблиці ніяких дій зі збереження посилальної цілісності даних не робиться.

6. Оновлення первинного ключа в рядку батьківської таблиці. Якщо значення первинного ключа деякого рядка батьківської таблиці буде оновлено, порушення посилальної цілісності станеться за тієї умови, що в дочірньому відношенні існують рядки, що посилаються на початкове значення первинного ключа. Для збереження посилальної цілісності може застосовуватися будь-яка з описаних вище стратегій. При використанні стратегії CASCADE оновлення значення первинного ключа в рядку батьківської таблиці буде відображене у будь-якому рядку дочірньої таблиці, що посилається на цей рядок.

Існує і інший вид цілісності – смислова (семантична) цілісність бази даних. Вимога смислової цілісності визначає, що дані у базі даних повинні змінюватися так, щоб не порушувався смисловий зв'язок, що склався між ними.

Рівень підтримки цілісності даних в різних системах істотно варіюється.

Ідеологія архітектури клієнт-сервер вимагає перенесення максимально можливого числа правил цілісності даних на сервер. До переваг такого підходу відносяться:

• гарантія цілісності бази даних, оскільки усі правила зосереджені в одному місці (у базі даних);

• автоматичне застосування визначених на сервері обмежень цілісності для будь-яких застосувань;

• відсутність різних реалізацій обмежень в різних клієнтських застосуваннях, працюючих з базою даних;

• швидке спрацьовування обмежень, оскільки вони реалізовані на сервері і, отже, немає необхідності посилати дані клієнтові, збільшуючи при цьому мережевий трафік;

• доступність внесених в обмеження на сервері змін для усіх клієнтських застосувань, працюючих з базою даних, і відсутність необхідності повторного поширення змінених додатків клієнтів серед користувачів.

До недоліків зберігання обмежень цілісності на сервері можна віднести:

• відсутність у клієнтського додатка можливості реагувати на деякі помилкові ситуації, що виникають на сервері при реалізації тих або інших правил (наприклад, помилок при виконанні процедур, що зберігаються, на сервері);

• обмеженість можливостей мови SQL і мови процедур, що зберігаються, і тригерів для реалізації усіх виникаючих потреб визначення цілісності даних.

На практиці в клієнтських застосунках реалізують лише такі правила, які важко або неможливо реалізувати із застосуванням засобів сервера. Усі інші обмеження цілісності даних переносяться на сервер.

Вимоги замовника (або корпоративні обмеження цілісності, або вимоги конкретного підприємства) передбачають існування додаткових правил підтримки цілісності даних, що визначаються користувачами, зовнішнім середовищем, прийняті на підприємстві або адміністраторами баз даних. Можуть вимагати застосування специфічних засобів контролю, зокрема тригерів.

1.2 Визначення обмежень цілісності

При створенні баз даних велика увага має бути приділена засобам підтримки даних в цілісному стані. Розглянемо передбачені стандартом мови SQL функції, які призначені для підтримки цілісності даних. Ця підтримка включає засоби завдання обмежень, вони вводяться з метою захисту бази даних від порушення узгодженості що зберігаються в ній даних. Велика частина цих обмежень задається в операторах CREATE TABLE і ALTER TABLE.

У стандарті SQL дано декілька варіантів визначення оператора створення таблиці, проте його базовий формат має наступний вигляд:

```
<визначення_таблиці> ::=
CREATE TABLE ім'я_таблиці
{(ім'я_стовпця тип_даних [ NOT NULL ][ UNIQUE]
[DEFAULT <значення>]
[ CHECK (<умова_вибору>)][,..n]}
[CONSTRAINT ім'я_обмеження]
[PRIMARY KEY (ім'я_стовпця [,..n])
{[UNIQUE (ім'я_стовпця [,..n])}
[FOREIGN KEY (ім'я_стовпця_зовнішнього_ключа
    [,..n])
```

```
REFERENCES ім'я_рід_таблиці
      [(ім'я_стовпця_рід_таблиці [,..n])],
[MATCH {PARTIAL | FULL}]
[ON UPDATE {CASCADE| SET NULL |SET DEFAULT
      |NO ACTION}]
[ON DELETE {CASCADE| SET NULL |SET DEFAULT
            |NO ACTION}]
{[CHECK(<умова_вибору>)][,..n]})
```

Представлена версія оператора створення таблиці включає засоби визначення вимог цілісності даних, а також інші конструкції. Є дуже великі варіації в наборі функціональних можливостей цього оператора, реалізованих в різних діалектах мови SQL. Розглянемо призначення параметрів команди, використовуваних для підтримки цілісності даних.

Обов'язкові дані. Для деяких стовпців потрібно наявність в кожному рядку таблиці конкретного і допустимого значення, відмінного від опущеного значення або значення NULL. Для завдань обмежень подібного типу стандарт SQL передбачає використання специфікації NOT NULL.

Вимоги конкретного підприємства. Оновлення даних в таблицях можуть бути обмежені існуючими в організації вимогами (бізнес-правилами). Стандарт SQL дозволяє реалізувати бізнес-правила підприємств за допомогою пропозиції CHECK і ключового слова UNIQUE.

Обмеження для доменів полів. Кожен стовпець має власний домен – деякий набір допустимих значень. Стандарт SQL передбачає два різні механізми визначення доменів. Перший полягає у використанні пропозиції CHECK, що дозволяє задати необхідні обмеження для стовпця або таблиці в цілому, а другою припускає застосування оператора CREATE DOMAIN.

Цілісність сутностей. Первинний ключ таблиці повинен мати унікальне не порожнє значення в кожному рядку. Стандарт SQL дозволяє задавати подібні вимоги підтримки цілісності даних за допомогою фрази PRIMARY KEY. В межах таблиці вона може вказуватися тільки один раз.

Проте існує можливість гарантувати унікальність значень і для будь-яких альтернативних ключів таблиці, що забезпечує ключове слово UNIQUE. Крім того, при визначенні альтернативних ключів рекомендується використовувати і специфікатори NOT NULL.

Посилальна цілісність. Зовнішні ключі є стовпцями або наборами стовпців, призначеними для зв'язування кожної з рядків дочірньої таблиці, що містить цей зовнішній ключ, з рядком батьківської таблиці, що містить відповідне значення потенційного ключа. Стандарт SQL передбачає механізм визначення зовнішніх ключів за допомогою пропозиції FOREIGN KEY, а фраза REFERENCES визначає ім'я батьківської таблиці, тобто таблиці, де знаходиться відповідний потенційний ключ. При використанні цієї пропозиції система відхилить виконання будь-яких операторів INSERT або UPDATE, за допомогою яких буде зроблена спроба створити в дочірній таблиці значення зовнішнього ключа, що не відповідає одному із вже існуючих значень потенційного ключа батьківської таблиці. Коли дії системи виконуються при вступі операторів UPDATE і DELETE, що містять спробу відновити або видалити значення потенційного ключа у батьківській таблиці, якому відповідає одна або більше за рядки дочірньої таблиці, то вони залежать від правил підтримки посилальної цілісності, вказаних у фразах ON UPDATE і ON DELETE пропозиції FOREIGN KEY. Якщо користувач робить спробу видалити з батьківської таблиці рядок, на який посилається одна або більше за рядки дочірньої таблиці, мова SQL надає наступні можливості:

- CASCADE – виконується видалення рядка з батьківської таблиці, що супроводжується автоматичним видаленням усіх рядків дочірньої таблиці, що посилаються на неї;
- SET NULL – виконується видалення рядка з батьківської таблиці, а в зовнішні ключі усіх рядків дочірньої таблиці, що посилаються на неї, записується значення NULL;

- SET DEFAULT – виконується видалення рядка з батьківської таблиці, а в зовнішні ключі усіх рядків дочірньої таблиці, що посилаються на неї, заноситься значення, що приймається за умовчанням;
- NO ACTION – операція видалення рядка з батьківської таблиці відміняється. Саме це значення використовується за умовчанням в тих випадках, коли в описі зовнішнього ключа фраза ON DELETE опущена.

Ті ж самі правила застосовуються в мові SQL і тоді, коли значення потенційного ключа батьківської таблиці оновлюється.

Визначник MATCH дозволяє уточнити спосіб обробки значення NULL в зовнішньому ключі.

При визначенні таблиці пропозиція FOREIGN KEY може вказуватися довільна кількість разів.

У операторові CREATE TABLE використовується необов'язкова фраза DEFAULT, яка призначена для задання по замовчуванні значення, коли в операторі INSERT значення в цьому стовпці буде відсутнє.

Фраза CONSTRAINT дозволяє задати ім'я обмеженню, що дозволить згодом відмінити те або інше обмеження за допомогою оператора ALTER TABLE.

Для внесення змін до вже створених таблиць стандартом SQL передбачений оператор ALTER TABLE, призначений для виконання наступних дій:

- додавання в таблицю нового стовпця;
- видалення стовпця з таблиці;
- додавання у визначення таблиці нового обмеження;
- видалення з визначення таблиці існуючого обмеження;
- завдання для стовпця значення за умовчанням;
- відміна для стовпця значення за умовчанням.

Оператор зміни таблиці має наступний узагальнений формат:

```
<зміна_таблиці> ::=
ALTER TABLE ім'я_таблиці
[ADD [COLUMN]ім'я_стовпця тип_даних
                [ NOT NULL ][UNIQUE]
[DEFAULT <значення>][ CHECK (<умова_вибору>)]]
[DROP [COLUMN] ім'я_стовпця [RESTRICT | CASCADE ]]
[ADD [CONSTRAINT [ім'я_обмеження]]
[{PRIMARY KEY (ім'я_стовпця [,..n])
      |[UNIQUE (ім'я_стовпця [,..n])}
|[FOREIGN KEY (ім'я_стовпця_зовнішнього_ключа [,..n])
      REFERENCES ім'я_рід_таблиці
          [(ім'я_стовпця_рід_таблиці [,..n])],
[ MATCH {PARTIAL | FULL}
      [ON UPDATE {CASCADE| SET NULL |
           SET DEFAULT | NO ACTION}]
      [ON DELETE {CASCADE| SET NULL |
           SET DEFAULT | NO ACTION}]
      |[CHECK(<умова_вибору>)][,..n]}]
[DROP CONSTRAINT ім'я_обмеження
       [RESTRICT | CASCADE]]
[ALTER [COLUMN] SET DEFAULT <значення>]
[ALTER [COLUMN] DROP DEFAULT]
```

Тут параметри мають те ж саме призначення, що і у визначенні оператора CREATE TABLE.

Оператор ALTER TABLE реалізований не в усіх діалектах мови SQL. У деяких діалектах він підтримується, проте не дозволяє видаляти з таблиці вже існуючі стовпці.

Для видалення таблиці використовується команда DROP TABLE.

1.3 Визначення обмежень цілісності в середовищі MS SQL Server

В процесі проектування бази даних приймається рішення про те, які таблиці повинні входити у базу даних, які у них будуть імена (ідентифікатори), які типи даних знадобляться для побудови таблиць і які користувачі отримають доступ до кожної з них. Крім того, для ефективного створення таблиць необхідно відповісти на наступні питання:

- Стовпці якого типу і розміру складатимуть кожну з таблиць, які вимагається вибрати імена для стовпців таблиць?
- Які стовпці можуть містити значення NULL?
- Чи будуть використані обмеження цілісності, значення за умовчанням і правила для стовпців?
- Чи потрібне індексування стовпців, які типи індексів будуть застосовані для конкретних стовпців?
- Які стовпці входитимуть в первинні і зовнішні ключі.

Для створення таблиць в середовищі MS SQL Server використовується команда:

<визначення_таблиці> ::=
CREATE TABLE [ім'я_бази_даних.[власник].
| власник.]ім'я_таблиці
((<елемент_таблиці>[,..n])
де
<елемент_таблиці> ::=
{<визначення_стовпця>}
| <ім'я_стовпця> AS <вираження>
| <обмеження_таблиці>

Зазвичай власником таблиці (dbo) є той, хто її створив.

<Вираження> задає значення для обчислюваного стовпця. Обчислювані стовпці - це віртуальні стовпці, т. е. фізично в таблиці вони не зберігаються і обчислюються з використанням значень стовпців тієї ж

таблиці. У вираженні для обчислюваного стовпця можуть бути присутніми імена звичайних стовпців, константи і функції, пов'язані одним або декількома операторами. Підзапити в такому вираженні брати участь не можуть. Обчислювані стовпці можуть бути включені в розділ SELECT при вказівці списку стовпців, які мають бути повернені в результаті виконання запиту. Обчислювані стовпці не можуть входити в зовнішній ключ, для них не використовуються значення за умовчанням. Крім того, обчислювані стовпці не можуть брати участь в операціях INSERT і DELETE.

<визначення_стовпця> ::=
{ ім'я_стовпця <тип_даних>}
[[DEFAULT <вираження>]
| [IDENTITY (початок, крок)[NOT FOR REPLICATION]]]]
[ROWGUIDCOL][<обмеження_стовпця>][..n]]

У визначенні стовпця звернемо увагу на параметр IDENTITY, який вказує, що відповідний стовпець буде стовпцем-лічильником. Для таблиці може бути визначений тільки один стовпець з такою властивістю. Можна додатково вказати початкове значення і крок приросту. Якщо ці значення не вказуються, то за умовчанням вони обоє рівні 1. Якщо з ключовим словом IDENTITY вказане NOT FOR REPLICATION, то сервер не виконуватиме автоматичного генерування значень для цього стовпця, а дозволить вставку в стовпець довільних значень.

В якості обмежень використовуються обмеження стовпця і обмеження таблиці. Відмінність між ними в тому, що обмеження стовпця застосовується тільки до певного поля, а обмеження таблиці - до груп з одного або більше за поля.

<обмеження_стовпця>::=
[CONSTRAINT ім'я_обмеження]
{ [NULL | NOT NULL]

```
| [ {PRIMARY KEY | UNIQUE }
[ CLUSTERED | NONCLUSTERED ]
[ WITH FILLFACTOR=чинник_заповнення ]
[ ON {ім'я_групи_файлів | DEFAULT } ] ] ]
| [ [ FOREIGN KEY ]
REFERENCES ім'я_рід_таблиці
      [(ім'я_стовпця_рід_таблиці) ]
[ ON DELETE { CASCADE | NO ACTION } ]
[ ON UPDATE { CASCADE | NO ACTION } ]
[ NOT FOR REPLICATION ]]
| CHECK [ NOT FOR REPLICATION](<балка_вираження>) }

<обмеження_таблиці>::=
[CONSTRAINT ім'я_обмеження ]
{ [ {PRIMARY KEY | UNIQUE }
      [ CLUSTERED | NONCLUSTERED ]
{(ім'я_стовпця [ASC | DESC][,..n])}
[WITH FILLFACTOR=чинник_заповнення ]
[ON {ім'я_групи_файлів | DEFAULT } ]]
|FOREIGN KEY[(ім'я_стовпця [,..n])]
REFERENCES ім'я_рід_таблиці
      [(ім'я_стовпця_рід_таблиці [,..n])]
[ ON DELETE { CASCADE | NO ACTION } ]
[ ON UPDATE { CASCADE | NO ACTION } ]
| NOT  FOR REPLICATION ]
| CHECK [ NOT FOR REPLICATION ] (балка_вираження)}
```

Розглянемо окремі параметри представлених конструкцій, пов'язані з обмеженнями цілісності даних. Обмеження цілісності мають пріоритет над тригерами, правилами і значеннями за умовчанням. До обмежень цілісності відносяться обмеження первинного ключа PRIMARY KEY,

обмеження зовнішнього ключа FOREIGN KEY, обмеження унікальності UNIQUE, обмеження значення NULL, обмеження на перевірку CHECK .

Обмеження первинного ключа (PRIMARY KEY). Таблиця зазвичай має стовпець або комбінацію стовпців, значення яких унікально ідентифікують кожен рядок в таблиці. Цей стовпець (чи стовпці) називається первісним ключем таблиці і потрібний для забезпечення її цілісності. Якщо в первинний ключ входить більше за один стовпець, то значення в межах одного стовпця можуть дублюватися, але будь-яка комбінація значень усіх стовпців первинного ключа має бути унікальна.

При створенні первинного ключа SQL Server автоматично створює унікальний індекс для стовпців, що входять в первинний ключ. Він прискорює доступ до даних цих стовпців при використанні первинного ключа в запитах.

Таблиця може мати тільки одно обмеження PRIMARY KEY, причому жоден з включених в первинний ключ стовпців не може набувати значення NULL. При спробі використовувати як первинний ключ стовпець (чи групу стовпців), для якого обмеження первинного ключа не виконуються, первинний ключ створений не буде, а система видасть повідомлення про помилку.

Оскільки обмеження PRIMARY KEY гарантує унікальність даних, воно часто визначається для стовпців-лічильників. Створення обмеження цілісності PRIMARY KEY можливо як при створенні, так і при зміні таблиці. Одним з призначень первинного ключа є забезпечення посилальної цілісності даних декількох таблиць. Природно, це може бути реалізовано тільки при визначенні відповідних зовнішніх ключів в інших таблицях.

Обмеження зовнішнього ключа (FOREIGN KEY). Обмеження зовнішнього ключа – це основний механізм для підтримки посилальної цілісності між таблицями реляційної бази даних. Стовпець дочірньої таблиці, визначений в якості зовнішнього ключа в параметрі FOREIGN KEY, застосовується для посилання на стовпець батьківської таблиці, що є

в ній первинним ключем. Ім'я батьківської таблиці і стовпці її первинного ключа вказуються в пропозиції REFERENCES. Дані в стовпцях, визначених в якості зовнішнього ключа, можуть приймати тільки такі ж значення, які знаходяться в пов'язаних з ним стовпцях первинного ключа батьківської таблиці. Збіг імен стовпців для зв'язку дочірньої і батьківської таблиць необов'язково. Первинний ключ може бути визначений для стовпця з одним ім'ям, тоді як стовпець, на який накладено обмеження FOREIGN KEY, може мати абсолютно інше ім'я. Єдиною вимогою залишається відповідність стовпців за типом і розміром даних.

На первинний ключ можуть посилатися не лише стовпці інших таблиць, але і стовпці, розташовані в тій же таблиці, що і власне первинний ключ ; це дозволяє створювати рекурсивні структури.

Зовнішній ключ може бути пов'язаний не лише з первинним ключем іншої таблиці. Він може бути визначений і для стовпців з обмеженням UNIQUE другої таблиці або будь-яких інших стовпців, але таблиці повинні знаходитися в одній базі даних.

Стовпці зовнішнього ключа можуть містити значення NULL, проте перевірка на обмеження FOREIGN KEY ігнорується. Зовнішній ключ може бути проіндексований, тоді сервер швидше відшукуватиме потрібні дані. Зовнішній ключ визначається як при створенні, так і при зміні таблиць.

Обмеження посилальної цілісності задає вимогу, згідно з якою для кожного запису в дочірній таблиці має бути запис у батьківській таблиці. При цьому зміна значення стовпця зв'язку в записі батьківської таблиці за наявності дочірнього запису блокується, так само як і видалення батьківського запису (заборона каскадної зміни і видалення), що гарантується параметрами ON DELETE NO ACTION і ON UPDATE NO ACTION, прийнятими за умовчанням. Для дозволу каскадної дії слід використовувати параметри ON DELETE CASCADE і ON UPDATE CASCADE.

Обмеження унікального ключа (UNIQUE). Це обмеження задає вимогу унікальності значення поля (стовпця) або групи полів (стовпців),

що входять в унікальний ключ, по відношенню до інших записів. Обмеження UNIQUE для стовпця таблиці схоже на первинний ключ: для кожного рядка даних в нім повинні міститися унікальні значення. Встановивши для деякого стовпця обмеження первинного ключа, можна одночасно встановити для іншого стовпця обмеження UNIQUE. Відмінність в обмеженні первинного і унікального ключа полягає в тому, що первинний ключ служить як для впорядкування даних в таблиці, так і для з'єднання пов'язаних між собою таблиць. Крім того, при використанні обмеження UNIQUE допускається існування значення NULL, але лише єдиний раз.

Обмеження на порожнє значення (NOT NULL). Для кожного стовпця таблиці можна встановити обмеження NOT NULL, що забороняє введення в цей стовпець нульового значення.

Обмеження перевірочне (CHECK) і правила. Це обмеження використовується для перевірки допустимості даних, що вводяться в конкретний стовпець таблиці, тобто обмеження CHECK забезпечує ще один рівень захисту даних.

Обмеження цілісності CHECK задають діапазон можливих значень для стовпця або стовпців. У основі обмежень цілісності CHECK лежить використання логічних виразів.

Допускається застосування декількох обмежень CHECK до одного і тому ж стовпця. В цьому випадку вони будуть застосовні в тій послідовності, в якій відбувалося їх створення. Можливе застосування одного і того ж обмеження до різних стовпців і використання в логічних виразах значень інших стовпців. Вказівка параметра NOT FOR REPLICATION наказує не виконувати перевірочних дій, якщо вони виконуються підсистемою реплікації.

Перевірочні обмеження можуть бути реалізовані і за допомогою правил. Правило є самостійним об'єктом бази даних, який зв'язується із стовпцем таблиці або призначеним для користувача типом даних. Причому одно і те ж правило може бути одночасно пов'язане з декількома

стовпцями і призначеними для користувача типами даних, що є безперечною перевагою. Проте істотний недолік полягає в тому, що з кожним стовпцем або типом даних може бути пов'язано тільки одне правило. Дозволяється комбінування обмежень цілісності CHECK з правилами. В цьому випадку виконується перевірка відповідності значення, що вводиться, як обмеженням цілісності, так і правилам.

Правило може бути створене командою:

CREATE RULE ім'я_правила AS вираження

Щоб зв'язати правило з тим або іншим стовпцем якої-небудь таблиці, необхідно використовувати системну процедуру, що зберігається :

sp_bindrule [@rulename=] 'rule'
[@objname=] 'object_name'
[,[@futureonly=['futureonly_flag']
Щоб відмінити правила, слід виконати наступну процедуру:
sp_unbindrule [@objname=] 'object_name'
[,[@futureonly=['futureonly_flag']
Видалення правила робиться командою
DROP RULE {ім'я_правила} [,..n].

Обмеження за умовчанням (DEFAULT). Стовпцю може бути присвоєне значення за умовчанням. Воно буде актуальним у тому випадку, якщо користувач не введе в стовпець ніякого іншого значення.

Окремо необхідно відмітити користь від використання значень за умовчанням при додаванні нового стовпця в таблицю. Якщо для стовпця, що додається, не дозволено зберігання значень NULL і не визначено значення за умовчанням, то операція додавання стовпця закінчиться невдачею.

При визначенні в таблиці стовпця з параметром ROWGUIDCOL сервер автоматично визначає для нього значення за умовчанням у вигляді функції NEWID(). Таким чином, для кожного нового рядка автоматично генеруватиметься глобальний унікальний ідентифікатор.

Додатковим механізмом використання значень за умовчанням є об'єкти бази даних, створені командою:

CREATE DEFAULT ім'я_умовчання AS константа

Умовчання зв'язується з тим або іншим стовпцем якої-небудь таблиці за допомогою процедури:

sp_bindefault [@defname=] 'default
[@objname=] 'object_name'
[,[@futureonly=] 'futureonly_flag']
де
"object_name'
може бути представлений як
"ім'я_таблиці.ім'я_стовпця'

Видалення обмеження за умовчанням виконується командою

DROP DEFAULT {ім'я_умовчання} [,..n]

якщо заздалегідь це обмеження було видалене з усіх таблиць процедурою

sp_unbindefault [@objname=] 'object_name'
[,[@futureonly=] 'futureonly_flag']

При створенні таблиці, окрім розглянутих прийомів, можна вказати необов'язкове ключове слово CONSTRAINT, щоб присвоїти обмеженню ім'я, унікальне в межах бази даних.

Ключові слова CLUSTERED і NONCLUSTERED дозволяють створити для стовпця кластерний або некластерний індекс. Для обмеження PRIMARY KEY за умовчанням створюється кластерний індекс, а для обмеження UNIQUE - некластерний. У кожній таблиці може бути створений лише один кластерний індекс, відмітною особливістю якого є те, що відповідно до нього змінюється фізичний порядок рядків в таблиці. ASC і DESC визначають метод впорядкування даних в індексі.

За допомогою параметра WITH FILLFACTOR=чинник_заповнення задається міра заповнення індексних сторінок при створенні індексу. Значення чинника заповнення вказується у відсотках і може змінюватися в проміжку від 0 до 100.

Параметр ON ім'я_групи_файлів означає групу, в якій передбачається зберігати таблицю.

Зміни в таблицю можна внести командою:

```
<зміна_таблиці> ::=
ALTER TABLE ім'я_таблиці
{[ALTER COLUMN ім'я_стовпця
{ тип_даних [(точність[,масштаб])]
[ NULL | NOT NULL ]
| {ADD | DROP } ROWGUIDCOL }]
| ADD { [<визначення_стовпця>]
| ім'я_стовпця AS вираження } [,..n]
| [WITH CHECK | WITH NOCHECK ]
        ADD  { <обмеження-таблиці> } [,..n]
| DROP
{ [CONSTRAINT ] ім'я_обмеження
```

| COLUMN ім'я_стовпця}[,..n]
| {CHECK | NOCHECK } CONSTRAINT
{ALL | ім'я_обмеження[,..n]}
| {ENABLE | DISABLE } TRIGGER
{ALL | ім'я_тригера [,..n]}}

На додаток до вже названих параметрів визначимо параметр {ENABLE | DISABLE } TRIGGER ALL, який дозволяє задіяти або відключити конкретний тригер або усі тригери, пов'язані з таблицею.

Видалення таблиці виконується командою:

DROP TABLE ім'я_таблиці

Видалити можна будь-яку таблицю, навіть системну. До цього питання треба підходити дуже обережно. Проте видаленню не підлягають таблиці, якщо існують об'єкти, що посилаються на них. До таких об'єктів відносяться таблиці, пов'язані з таблицею, що видаляється, за допомогою зовнішнього ключа. Тому, перш ніж видаляти батьківську таблицю, необхідно видалити або обмеження зовнішнього ключа, або дочірні таблиці. Якщо з таблицею пов'язано хоч би одно представлення, то таблицю також видалити не вдасться. Крім того, зв'язок з таблицею може бути встановлений з боку функцій і процедур. Отже, перед видаленням таблиці необхідно видалити усі об'єкти бази даних, які на неї посилаються, або змінити їх так, щоб посилань на таблицю, що видаляється, не було.

2 ВИКОНАННЯ РОБОТИ

При виконанні лабораторної роботи передбачається використання застосунку SQL Server Management Studio, який входить до складу інтегрованої платформи Microsoft SQL Server. При цьому передбачається використання СУБД Microsoft SQL Server версії 2008 або вищої. У зв'язку із цим елементи інтерфейсу можуть мати відмінності порівняно із тими, що наведено далі у вигляді рисунків (особливо у випадку використання версії, локалізованої для використання певної національної мови). Ці відмінності не є принциповими та не впливають на виконання роботи та отримані результати.

Увага! При виконанні цієї лабораторної роботи, рекомендується використовувати окрему базу даних. Її можна спеціально створити, використовуючи запити, створені при виконанні лабораторної роботи за темою «Ознайомлення з основними особливостями СУБД Microsoft SQL Server. Створення бази даних та об'єктів бази даних».

Також треба звернути увагу на те, що далі будуть розглянуті питання, пов'язані із використанням засобів контролю посилальної цілісності. Це обумовлено тим, що інші види обмежень цілісності даних та засоби їх підтримки (наприклад, обов'язкові дані, тригери ті інші), вже розглядалися у попередніх лабораторних роботах.

2.1 Вивчення особливостей роботи механізму контролю посилальної цілісності No Action

Розглянемо особливості роботи механізму посилальної цілісності No Action на прикладі відносин між таблицями «Suppliers» та «Contracts», «Suppliers» та «IndividualEntrepreneurs», «Suppliers» та «LegalEntities». Ці таблиці пов'язані між собою по полю SupplierID. У цих зв'язках таблиця «Suppliers» є батьківською, а таблиці «Contracts», «LegalEntities», «IndividualEntrepreneurs» – дочірніми. Для вивчення особливостей роботи механізму цілісності посилань виконаємо наступну послідовність дій.

1. У списку таблиць вибрати таблицю «Suppliers», клацнувши по ній правою кнопкою миші. У меню вибрати пункт Modify. В результаті буде отримано доступ до редагування структури таблиці.

2. Клацнути правою кнопкою миші по будь-якому полю таблиці і в меню вибрати пункт Relationships ... В результаті на екрані з'явиться вікно Foreign Key Relationships (рисунок 2.1).

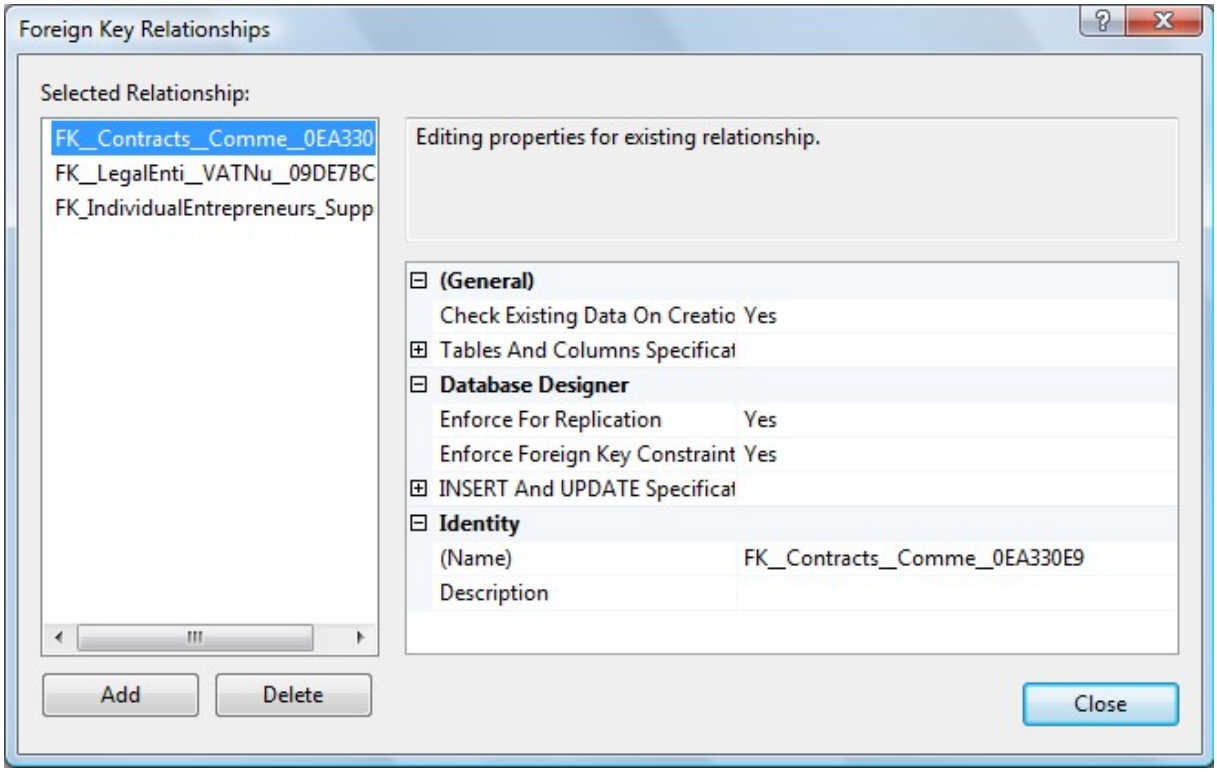


Рисунок 2.1

3. Вибрати зв'язок, що відповідає зв'язку між таблицями «Suppliers» та «Contracts» (рисунок 2.1). Розкривши пункт Tables And Columns Specification, можна побачити інформацію, що показує, які таблиці пов'язані і які ключі при цьому були використані.

4. Далі перейдемо до розгляду механізмів цілісності посилання, що використовуються при видаленні записів у таблиці «Suppliers» або зміни ключового значення (тобто значення поля SupplierID). Розкриємо пункт INSERT And UPDATE Specification (рисунок 2.2). Як видно, механізм цілісності посилання No Action встановлений за умовчанням. Так само

потрібно перевірити механізм посилальної цілісності для зв'язків між таблицями «Suppliers» та «IndividualEntrepreneurs» та «Suppliers» та «LegalEntities». Вікно Foreign Key Relationships необхідно закрити. Також необхідно закрити вікно, що забезпечує доступ до структури таблиці «Suppliers».

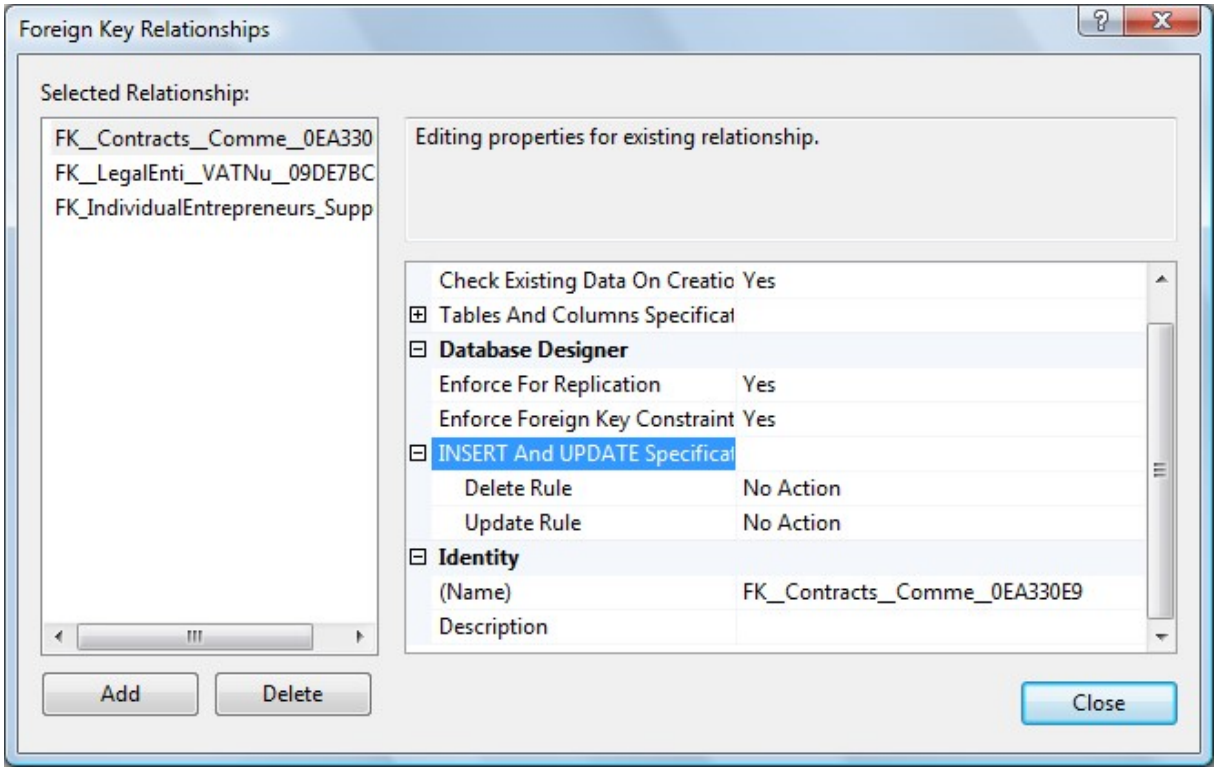


Рисунок 2.2

5. Відкрити таблицю «Suppliers» в режимі перегляду/редагування даних. Аналогічно потрібно відкрити таблиці «Contracts», «IndividualEntrepreneurs» та «LegalEntities».

6. Припустимо, що через якісь причини необхідно видалити постачальника з кодом 4. Вибравши відповідний запис у таблиці Постачальники, натисніть праву кнопку миші і в меню виберіть Delete. Потім підтвердьте видалення запису. Після цього на екран буде виведено вікно (рисунок 2.3), що інформує користувача про те, що видалення запису неможливе, тому що на цей запис посилаються записи у зв'язаних таблицях.

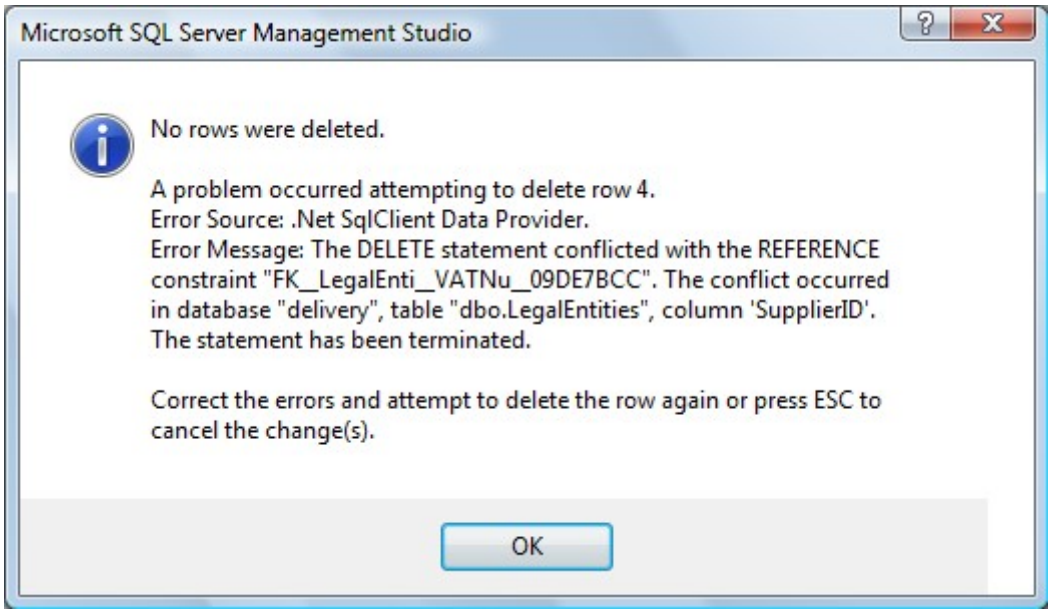


Рисунок 2.3

7. Таким чином, щоб видалити даного постачальника, потрібно попередньо видалити всі пов'язані з ним дані. Для цього потрібно видалити відповідний запис з таблиці «LegalEntities» та перевірити наявність договорів із цим постачальником у таблиці «Contracts». Якщо такі договори є, їх теж потрібно видалити (при цьому потрібно мати на увазі, що може виникнути потреба у видаленні та вмісті цих договорів). Після цього потрібно спробувати повторити спробу видалення постачальника з кодом 4. Якщо зв'язаних із ним даних немає, постачальника буде видалено.

8. Припустимо, що з якихось причин виникла потреба для постачальника з кодом 5 змінити код на 7. Вибравши відповідний запис у таблиці «Suppliers», змініть код постачальника з 5 на 7. Потім спробуйте перейти на попередній запис. Після цього на екран буде виведено вікно (рисунок 2.4), що інформує користувача про те, що зміна даних неможлива, тому що на цей код посилаються записи у зв'язаних таблицях. Оскільки договори з цим постачальником відсутні, посилання на нього є лише у таблиці «IndividualEntrepreneurs». Видаливши цей запис, повторіть спробу зміни коду постачальника з 5 на 7. Тепер ця операція повинна

пройти успішно. Після цього потрібно перевірити вміст таблиць та таблиці закрити.

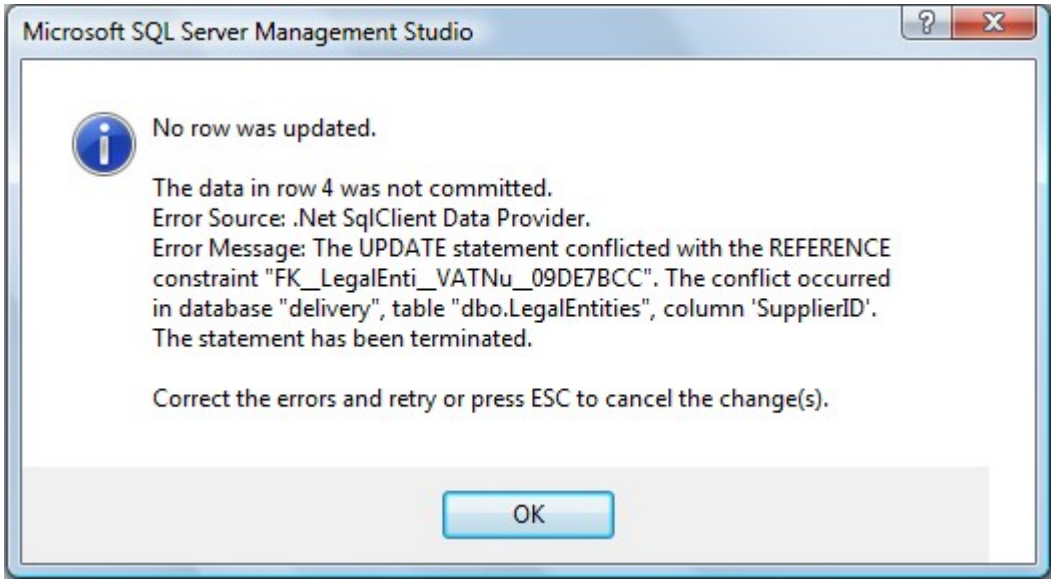


Рисунок 2.4

2.2 Вивчення особливостей роботи механізму контролю посилальної цілісності Cascade

Розглянемо особливості роботи механізму посилальної цілісності Cascade на прикладі відносин між таблицями «Suppliers» та «Contracts», «Suppliers» та «IndividualEntrepreneurs», «Suppliers» та «LegalEntities», «Contracts» та «Supplied». Доступ до зв'язків виконується так, як описано вище. Для вивчення особливостей роботи механізму цілісності посилань виконаємо наступну послідовність дій.

1. Змінимо механізми цілісності посилань для зв'язків між усіма таблицями на Cascade. Приклад результату зміни наведений на рисунку 2.5.

2. Припустимо, що з якихось причин виникла потреба для постачальника з кодом 2 змінити код на 8. Вибравши відповідний запис у таблиці Постачальники, змініть код постачальника з 2 на 8. Потім спробуйте перейти на попередній запис. Перевірте зміни значення код постачальника у безпосередньо пов'язаних з цим постачальником таблицях

(«LegalEntities», «Contracts»). Якщо зміни не з'явилися відразу, то таблицю потрібно закрити і потім знову відкрити або у вікні вмісту таблиці клацнути правою кнопкою миші і в меню вибрати пункт Execute SQL.

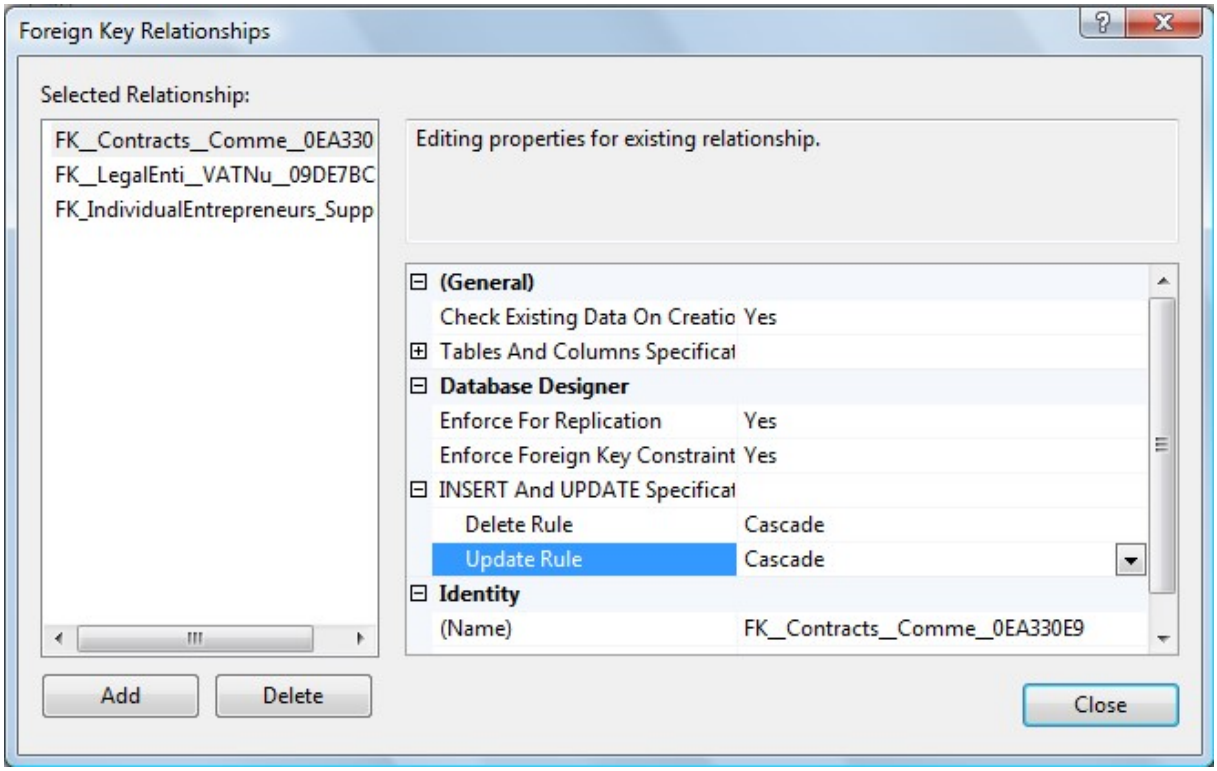


Рисунок 2.5

3. Тепер припустимо, що цього постачальника (який має код 8), необхідно видалити. Вибравши відповідний запис у таблиці «Suppliers», натисніть праву кнопку миші та в меню виберіть Delete. Потім підтвердьте видалення запису. Після цього перевірте стан даних у таблицях, які безпосередньо чи опосередковано пов'язані з цим постачальником («LegalEntities», «Contracts», «Supplied»). Переконайтеся, що дані видалено. Після цього потрібно таблиці закрити.

2.3 Вивчення особливостей роботи механізму контролю посилальної цілісності Set Null

Розглянемо особливості роботи механізму цілісності Set Null на прикладі відносин між таблицями «Suppliers» та «Contracts». Доступ до

зв'язків виконується так, як описано вище. Для вивчення особливостей роботи механізму цілісності посилань виконаємо наступну послідовність дій.

1. У списку таблиць вибрати таблицю «Contracts», клацнувши по ній правою кнопкою миші. У меню вибрати пункт Modify. В результаті буде отримано доступ до редагування структури таблиці. Для поля SupplierID встановити властивість Allow Nulls (рисунок 2.6).

Column Name	Data Type	Allow Nulls
ContractNumber	int	☐
ContractDate	datetime	☑
SupplierID	int	☑
ContractName	text	☑
Comment	text	☑
		☐

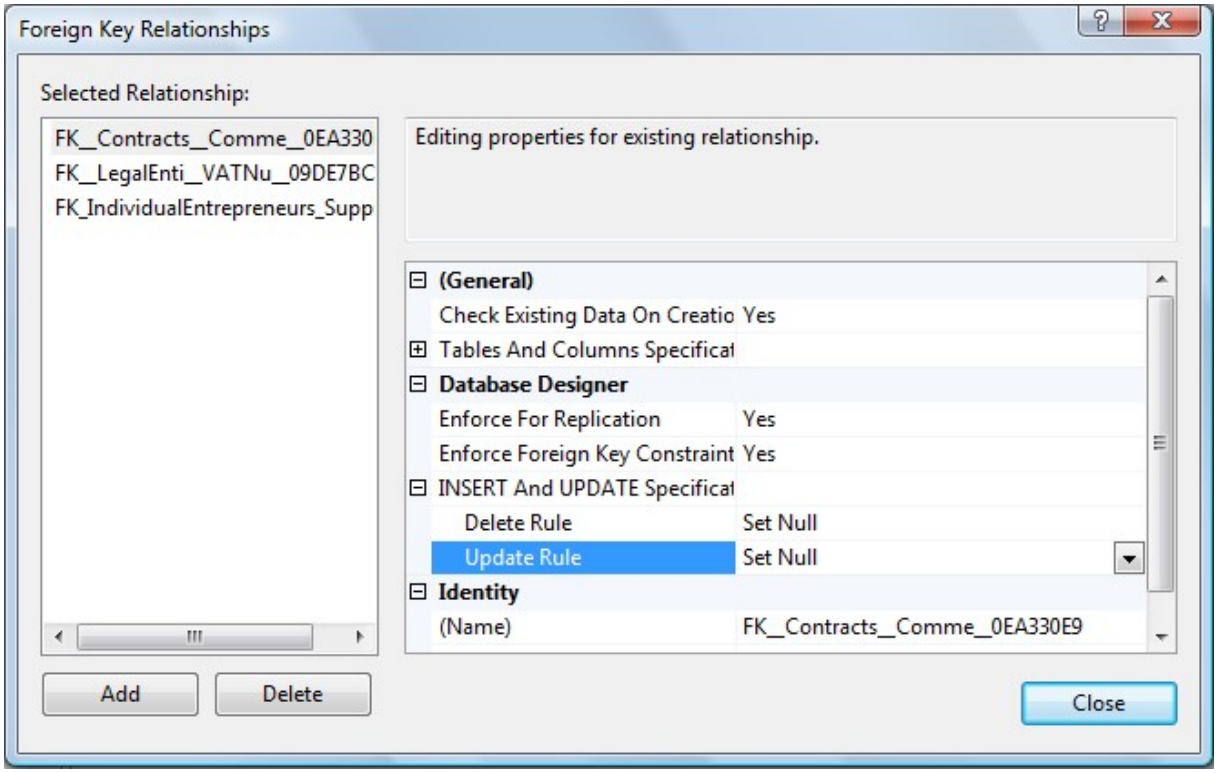
Рисунок 2.6

2. Клацнути правою кнопкою миші по будь-якому полю таблиці і в меню вибрати пункт Relationships ... В результаті на екрані з'явиться вікно Foreign Key Relationships. Змінити механізми цілісності посилання для зв'язку між таблицями «Suppliers» та «Contracts» на Set Null (рисунок 2.7). Зберегти зміни у таблиці.

3. Відкрити в режимі перегляду/модифікації даних таблиці «Suppliers» та «Contracts». Для договору 6 змінити код постачальника з 1 на 7. Потім у таблиці «Suppliers» змінити код постачальника 7 на 10. Перевірити дані у таблиці «Contracts». Код постачальника у договорі 6 повинен прийняти значення Null. Приклад таблиці із зміненими даними наведено рисунку 2.8.

4. У таблиці «Contracts» для договору 6 змініть код постачальника з Null на 10. Після цього в таблиці «Suppliers» видаліть постачальника з кодом 10. Перевірте стан даних у таблиці «Contracts». Для договору 6 значення коду постачальника знову має прийняти значення Null.

5. Відкрити для перегляду даних таблиці закрити.



	ContractNumber	ContractDate	SupplierID	ContractName	Comment
▶	1	1999-09-01 00:...	1	Договір № 1	Підстава - накл...
	2	1999-09-10 00:...	1	Договір № 2	Підстава - раху...
	3	1999-09-10 00:...	3	Договір № 3	Підстава - раху...
	4	1999-09-23 00:...	3	Договір № 4	Підстава - замо...
	5	1999-09-24 00:...	2	Договір № 5	Підстава - накл...
	6	1999-10-01 00:...	*NULL*	Договір № 6	Підстава - раху...
	7	1999-10-02 00:...	2	Договір № 7	Підстава - накл...
*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Рисунок 2.8

2.4 Вивчення особливостей роботи механізму контролю посилальної цілісності Set Default

Розглянемо особливості роботи механізму цілісності Set Default на прикладі відносин між таблицями «Suppliers» та «Contracts». Доступ до зв'язків виконується так, як описано вище. Для вивчення особливостей роботи механізму цілісності посилань виконаємо наступну послідовність дій.

1. Відкрити таблицю «Contracts» у режимі перегляду/модифікації даних. Для договору 6 змінити значення коду постачальника Null на 3. Закрити таблицю.

2. Відкрити таблицю «Contracts» у режимі редагування структури. Для поля SupplierID вимкнути властивість Allow Nulls (рисунок 2.9).

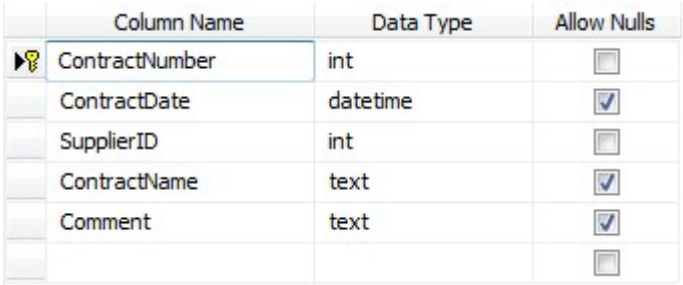


Рисунок 2.9

3. Вибрати поле SupplierID та встановити значення за умовчанням для цього поля. Для цього встановити для властивості Default Value or Binding значення 1 (рисунок 2.10).

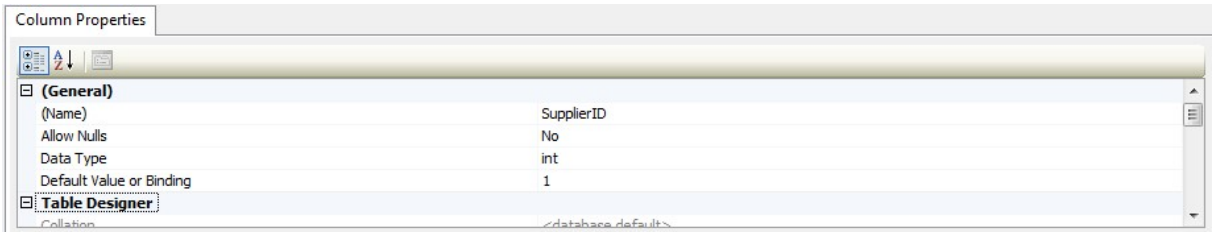


Рисунок 2.10

4. Клацнути правою кнопкою миші по будь-якому полю таблиці і в меню вибрати пункт Relationships…. На екрані з'явиться вікно Foreign Key Relationships. Змінити механізми цілісності посилання для зв'язку між таблицями Постачальники та Договори на Set Default (рисунок 2.11). Закрити вікно Foreign Key Relationships та зберегти зміни в таблиці.

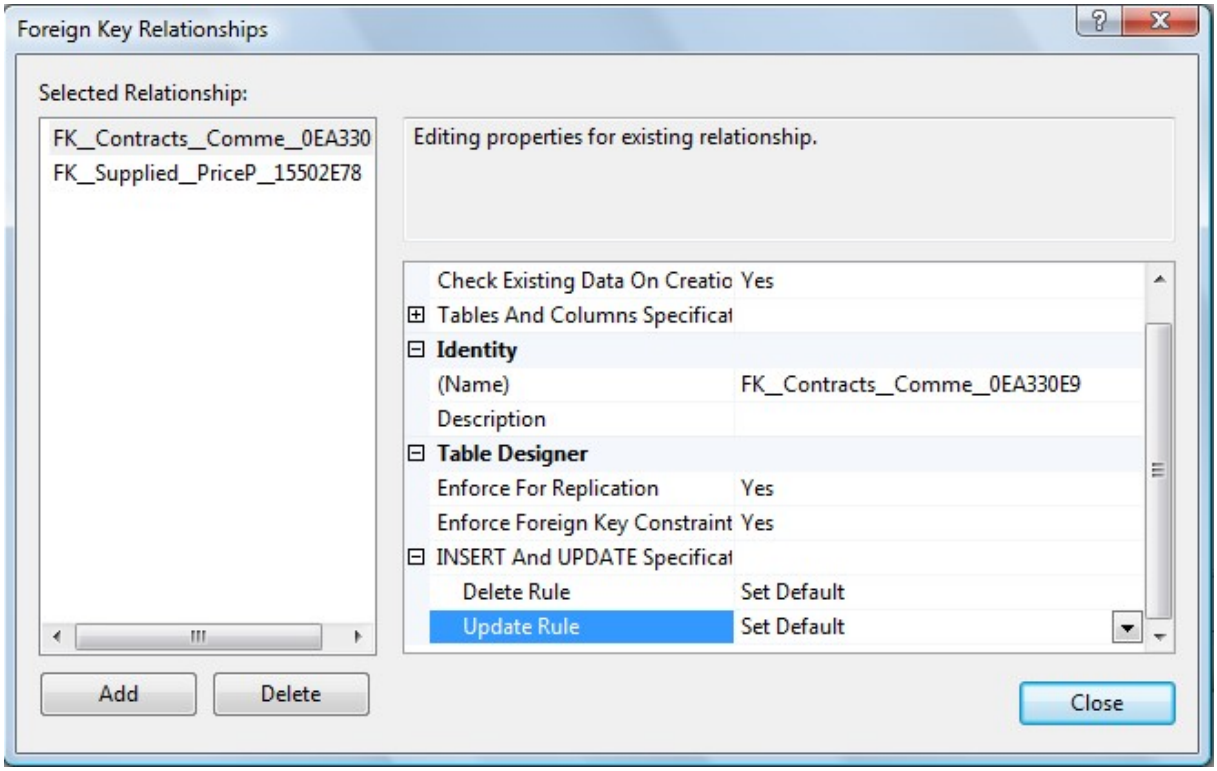


Рисунок 2.11

5. Відкрити в режимі перегляду даних таблиці «Suppliers» та «Contracts». У таблиці «Contracts» визначити список договорів, для яких код постачальника дорівнює 3. У таблиці «Suppliers» змінити код постачальника 3 на 12. Перевірити дані у таблиці «Contracts». Для договорів, для яких код постачальника дорівнював 3, код постачальника повинен змінитися на 1.

6. У таблиці «Contracts» змінити для деяких договорів (наприклад, для договорів 3, 4, 6) код постачальника з 1 до 12.

7. У таблиці «Suppliers» видалити запис, що відповідає постачальнику з кодом 12.

8. Перевірити дані у таблиці «Contracts». Для договорів, для яких код постачальника дорівнював 12, код постачальника повинен змінитися на 1.

Наведені вище приклади застосування механізмів контролю посилальної цілісності є досить простими. Більш детальну інформацію щодо контролю посилальної цілісності засобами мови Transact-SQL, можна, наприклад, отримати за посиланнями:

https://learn.microsoft.com/en-us/sql/relational-databases/tables/primary-and-foreign-key-constraints?view=sql-server-ver16

https://www.mullinsconsulting.com/sql_ref.htm

2.5 Збереження результатів роботи

Після закінчення роботи всі таблиці слід закрити, а потім базу даних відключити. Базу даних можна не зберігати.

3 ВИМОГИ ДО ЗВІТУ

Відповіднім чином оформлений та роздрукований звіт з лабораторної роботи є документом, що підтверджує виконання студентом лабораторної роботи.

У звіті з лабораторної роботи:

1) стисло описати основні етапи виконання завдання;

2) описати особливості розглянутих механізмів контролю посилальної цілісності та результати їх використання;

3) зробити висновки за результатами виконання лабораторної роботи.

Звіт з лабораторної роботи роздруковується на аркушах формату А4, він повинен мати відповідній титульний аркуш. Роздрукованій звіт здається студентом викладачу у файлі.

Звіт має буті оформлень за такими вимогами:

– параметри сторінки: лівий відступ – 3 см; правий – 1,5 см; верхній та нижній відступи по 2 см;

– шрифт Times New Roman, 14;

– налаштування абзацу: вирівнювання – за шириною, відступи зліва та справа – 0 см, відступ першого рядка - 1,25 см, інтервал перед та після абзацу – 0 пт, міжрядковій інтервал – одинарний; на вкладці «Положення на сторінці» відключити функцію «Заборона висячих рядків».

Усі скріншоти розміщені у звіті, є рисунками, отже повинні мати підписи та відповідну нумерацію.

Ознакою того, що студент не тільки виконав лабораторну роботу, але й здав її (тобто підтвердив наявність відповідних знань та навичок), є наявність на титульному аркуші підпису викладача та дати здачі.

4 ПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ

1. Наведіть основні типи обмежень цілісності даних.
2. Що таке обов'язкові дані?
3. Що таке обмеження для доменів полів?
4. Що таке корпоративні обмеження?
5. Що таке цілісність сутностей?
6. Що таке посилальна цілісність?
7. За допомогою яких ключів реалізується посилальна цілісність?
8. Що таке первинний ключ?
9. Що таке зовнішній ключ?
10. Що таке альтернативний ключ? Яким чином можна реалізувати альтернативний ключ?
11. Які різновиди зв'язку можуть існувати між таблицями бази даних?
12. Організація підтримки посилальної цілісності при виконанні операцій модифікації даних у базі. Які при цьому можливі ситуації?
13. Наведіть основні типи стратегій (або механізмів) контролю посилальної цілісності.
14. Стратегія (або механізм) контролю посилальної цілісності NO ACTION. Призначення та особливості використання.
15. Стратегія (або механізм) контролю посилальної цілісності CASCADE. Призначення та особливості використання.
16. Стратегія (або механізм) контролю посилальної цілісності SET NULL. Призначення та особливості використання.
17. Стратегія (або механізм) контролю посилальної цілісності SET DEFAULT. Призначення та особливості використання.
18. Переваги зберігання обмежень цілісності даних на сервері баз даних.
19. Недоліки зберігання обмежень цілісності даних на сервері баз даних.

20. Які обмеження цілісності даних задаються в операторі CREATE TABLE?

21. Які обмеження цілісності даних задаються в операторі ALTER TABLE?

22. Яким чином обов'язкові дані задаються в операторах CREATE TABLE та ALTER TABLE?

23. Яким чином обмеження посилальної цілісності задаються в операторах CREATE TABLE та ALTER TABLE?

24. Що таке правило? Як його можна реалізувати?

СПИСОК ЛІТЕРАТУРИ

1. Програмування баз даних. Практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 152 «Метрологія та інформаційно-вимірювальна техніка» / М. В. Добролюбова, М. В. Філіппова, О. М. Маркіна. – Київ : КПІ ім. Ігоря Сікорського, 2021. – 164 с.

2. Anthony DeBarros. Practical SQL. 2nd Edition. – No Starch Press, Inc., 2022. – 440 c.

3. Jeffrey A. Hoffer, V. Ramesh, Heikki Topi. Modern Database Management. Thirteenth Edition. – Pearson Education Limited, 2020. – 591 p.

4. Dušan Petković. Microsoft SQL Server 2019. A Beginner’s Guide. Seventh Edition. – McGraw-Hill Education, 2020. – 865 p.

5. Mukesh Negi. Fundamentals of Database Management System: Learn essential concepts of database systems. – BPB Publications, 2019. – 175 p.

6. Edward Sciore. Database Design and Implementation: Second Edition. – Springer Nature, 2020. – 468 p.

7. Gavin Powell. Database Modeling Step by Step. – CRC Press, 2020. – 268 p.

8. Sanjiv Purba. Handbook of Data Management: 1999 Edition. – CRC Press, 2019. – 1101 p.

9. C. J. Date. Database Design and Relational Theory: Normal Forms and All That Jazz. – Apress, 2019. – 451 p.

10. Jonathan Eckstein, Bonnie R. Schultz. Introductory Relational Database Design for Business, with Microsoft Access. – John Wiley & Sons, 2018. – 328 p.

11. Alan Beaulieu. Learning SQL: Generate, Manipulate, and Retrieve Data. – O'Reilly Media, Inc., 2020. – 384 p.

12. M. Tamer Özsu, Patrick Valduriez. Principles of Distributed Database Systems. – Springer Nature, 2019. – 674 p.

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторної роботи за темою «Вивчення основ роботи із засобами контролю цілісності даних засобами СУБД Microsoft SQL Server»
для студентів спеціальностей
121 «Інженерія програмного забезпечення»
122 «Комп’ютерні науки»
126 «Інформаційні системи та технології»

Укладачі:
ОРЛОВСЬКИЙ Дмитро Леонідович
КОПП Андрій Михайлович

Відповідальний за випуск Годлевський М.Д.
Роботу до видання рекомендував Гамаюн І.П.

План 2023 р., поз. 496

Підп. до друку __.__.__. Гарнітура Times New Roman.
Ум. друк. арк. 1,8.

Видавничий центр НТУ «ХПІ»,
вул. Кирпичова, 2, м. Харків, 61002
Свідоцтво про державну реєстрацію ДК № 3478 від 21.08.2017 р.

Самостійне електронне видання