**Лабораторна робота№4 Використання операторів визначення даних DDL.**

**Мета:** Набуття практичних навичок використання операторів визначення даних DDL

***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 лекцію №7. Ознайомитися з теоретичною частиною.
2. За структурою раніш створених таблиць Викладач, Студент, Група, Предмети, Розклад та Аудиторії написати SQL-запит на створення таблиць замінивши назву таблиць, запустити та порівняти структуру таблиць. Написати використані SQL-запити та при виявленні розбіжностей визначити їх.
3. До таблиць Викладач, Студент використовуючи команду ALTER TABLE додайте поля Код території проживання (код за КОАТУУ, числовий 10 знаків) . Написати використані SQL-запити.
4. Вилучити створені в п.2 таблиці командою DROP TABLE.
5. Встановити на раніш створених таблиці Викладач, Студент, Група, Предмети, Розклад та Аудиторії використовуючи команду ALTER TABLE обмеження відповідно до визначених в ПР№4 (первинні ключі, домени та ін..) Написати використані SQL-запити.
6. Сформувати звіт з наданням виконаних команд та відповідними поясненнями.
7. 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 - 05.10.2022**

***Контрольні запитання***

**ТЕОРЕТИЧНА ЧАСТИНА**

***Інструменти SQL для роботи таблицями***:

* команда CREATE TABLE створює нову таблицю;
* команда ALTER TABLE зміює структуру існуючої таблиці;
* команда DROP TABLE видаляє таблицю і всї її дані;
* команда CREATE TEMPORARY TABLE створює тимчасову таблицю, яка потім буде вилучена;
* команда SELECT INTO створює нову таблицю на основі вже існуючої.

На виконання цих команд ваша СУБД видасть повідомлення щодо виконання команди (виконана чи ні). Побачити результати можна переглянувши сзему БД у вашому інструментарії СУБД. Ці команди зміюють об’єкти бази даних (а в деяких випадках і самі дані), тому администратор БД може не дозволити їх використовувати.

***Порядок сстворення таблиць***

Щоб сстворити таблицю, потрібно вказати:

* назву таблиці;
* названу стовбців;
* тип даних для стовбців;
* обмеження

Назви таблиць та стовбців повинні відповідати вимогам SQL.

Для прикладу, розглянемо фрагмент бази даних “Книжковий магазин” (табл.1). Дана база даних призначена для обліку розповсюдження книг в книжному магазині. В даній базі міститься інформація про наявні книги в книжковому магазині, інформація про їх авторів та видавців, а також відомості про продажі та покупців (див. подальші таблиці).

Таблиця 1 Фрагмент бази даних “Книжковий магазин”

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Authors** | | **BookSales** | | **Buyers** | **Publishers** | **TitleAuthors** | **Titles** |
| AUID *(PK)* | | BookSalesID *(PK)* | | BuyerID *(PK)* | PubID *(PK)* | ISBN *(PK) (FK)* | ISBN *(PK)* |
| Surname | | Qtr | | Name | Name | AUID *(PK) (FK)* | Title |
| Name | Sales | | CompanyName | | Company | PubID *(FK)* | |
| Lastname | SalesRep | | Address | | Address | YearPub | |
| Address | ISBN *(FK)* | | City | | City | Description | |
| Date\_of\_birth | Units | | Region | | Region | Notes | |
| Contracted | BuyerID *(FK)* | | Zip | | Zip | Subject | |
| Phone | Phone | | | | | | Comments |
| Fax | Fax | | | | | | |
| Comments | | | | | | | |

Зв’язок між таблицями здійснюється за допомогою таких пар полів з типом зв’язку “один-до-багатьох” відповідно:

1. Authors.AUID – TitleAuthors.AUID

2. Titles.ISBN – TitleAuthors.ISBN

3. Publishers.PubID – Titles.PubID

4. Titles.ISBN – BookSalesID.ISBN

5. Buyers. BuyerID – BookSalesID. BuyerID

*Таблиця 2. Відомості про назви книг (таблиця Titles)*

|  |  |  |
| --- | --- | --- |
| **Назва поля** | **Тип поля** | **Опис поля** |
| ISBN | Integer | Універсальний ідентифікаійний номер книги |
| Title | Varchar | Назва книги |
| PubID | Integer | Ідентифікаійний номер видавництва |
| YearPub | Date | Дата видавництва |
| Description | Varchar | Короткий опис книги |
| Notes | Varchar | Примітки |
| Subject | Varchar | Тематика книги |
| Comments | Varchar | Коментарі |

*Таблиця 3. Список авторів (таблиця Authors)*

|  |  |  |
| --- | --- | --- |
| **Назва поля** | **Тип поля** | **Опис поля** |
| AUID | Integer | Ідентифікаійний номер автора |
| Surname | Varchar | Прізвище автора |
| Name | Varchar | Ім’я автора |
| Lastname | Varchar | По батькові автора |
| Address | Varchar | Адреса автора |
| Date\_of\_birth | Date | Дата народження |
| Contracted | Numeric | Номер договору |

*Таблиця 4. Таблиця, яка пов’язує Titles та Authors (таблиця TitleAuthors)*

|  |  |  |
| --- | --- | --- |
| **Назва поля** | **Тип поля** | **Опис поля** |
| ISBN | Integer | Універсальний ідентифікаійний номер книги |
| AUID | Integer | Ідентифікаійний номер автора |

Таблиця 5. Відомості про книжкові продажі (таблиця BookSales)

|  |  |  |
| --- | --- | --- |
| **Назва поля** | **Тип поля** | **Опис поля** |
| BookSalesID | Integer | Ідентифікаійний номер продажу |
| Qtr | Numeric | Квартал |
| Sales | Numeric | Ціна |
| SalesRep | Varchar | Ім’я торгового представника |
| ISBN | Integer | Ідентифікаійний номер книги |
| Units | Numeric | Кількість проданих книг |
| BuyerID | Integer | Ідентифікаійний номер покупця |

Таблиця 6. Відомості про покупців (таблиця Buyers)

|  |  |  |
| --- | --- | --- |
| **Назва поля** | **Тип поля** | **Опис поля** |
| BuyerID | Integer | Ідентифікаійний номер покупця |
| Surname | Varchar | Прізвище покупця |
| Name | Varchar | Ім’я покупця |
| Lastname | Varchar | По батькові покупця |
| СompanyName | Varchar | Назва компанії, яку представляє покупець |
| Address | Varchar | Адреса (вулиця, будинок, квартира) |
| City | Varchar | Місто |
| Region | Varchar | Область |
| Zip | Varchar | Поштовий код |
| Telephone | Varchar | Номер телефону |
| Fax | Varchar | Фах |

Таблиця 7. Відомості про видавців (таблиця Publishers)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Назва поля** | | **Тип поля** | | **Опис поля** |
| PubID | Integer | | Ідентифікаійний номер видавництва | |
| Name | Varchar | | Назва видавництва | |
| Сompany | Varchar | | Назва компанії | |
| Address | Varchar | | Адреса (вулиця, будинок, квартира) | |
| City | Varchar | | Місто | |
| Region | Varchar | | Регіон | |
| Zip | Varchar | | Поштовий код | |
| Phone | Varchar | | Номер телефону | |
| Fax | Varchar | | Фах | |
| Comments | Varchar | | Коментарі | |

**Створення таблиць**

Створення таблиці виконується за допомогою оператора ***CREATE TABLE*** (створити таблицю) із заданням необхідних параметрів. При цьому створюється постійна таблиця. Для створення тимчасової таблиці використовується оператор CREATE TEMPORARY TABLE (створити тимчасову таблицю). Тимчасова таблиця, на відміну від постійної, існує лише протягом сеансу роботи з базою даних, в якому вона була створена. Однак тимчасова таблиця може бути доступна іншим користувачам, як і постійна таблиця. Зазвичай тимчасові таблиці створюються для представлення в них поточних підсумкових (звітних) даних, доступних декільком користувачам бази даних. Далі наведено синтаксис оператора CREATE:

CREATE [TEMPORARY] TABLE Ім’яТаблиці (

{Стовпець1 тип [(розмір)] [обмеження\_стопця] [, ...]}

{[, CONSTRAINT обмеження\_таблиці] [,. . . ]}

);

Тут квадратні та фігурні дужки, на відміну від круглих, не є елементами синтаксису. У квадратних дужках задані необов’язкові елементи, а в фігурних – елементи, які можуть повторюватися.

Для створення таблиці необхідно вказати її ім’я та визначити стовпці. Визначення стовпця включає його ім’я і тип. Якщо вказується довжина стовпця, то вона береться в круглі дужки після типу. Крім того, можна вказати обмеження для стовпця. Всі перераховані елементи визначення стовпця задаються один за одним через пробіл. Якщо створювана таблиця містить кілька стовпців, то їх визначення розділяються комами. Обмеження може бути визначено і для всієї таблиці, а не тільки для її стовпців. У цьому випадку використовується ключове слово CONSTRAINT (обмеження), після якого вказується саме обмеження.

Далі наведено запит на створення простої таблиці без обмежень:

CREATE TABLE Titles (

ISBN CHAR(20),

Title VARCHAR,

PubID INTEGER,

YearPub DATE,

Description VARCHAR,

Notes VARCHAR,

Subject VARCHAR,

Comments VARCHAR

);

Даний запит створює таблицю Titles (див. табл. 2), що містить вісім стовпців.

Якщо вы спробуєте створити таблицю з ім’ям, яке вже існує в БД, СУБД видасть помилку. Для запобігання збереженню однієї таблиці замість іншої SQL вимагає, щоб перед створенням таблиця з таким же ім.’ям була вилучена за домогою команди DROP TABLE (див.далі).

Одразу після створення таблиця пуста (не має рядків). Для заповнення таблиці можна скористатися командою INSERT (наступні заняття)

**Обмеження для стовпців**

При створенні таблиці, ми можемо встановлювати обмеження на дані, які вводяться в таблицю. Наприклад, на числовий стовпець можна накласти обмеження, яке полягає в тому, що число повинне знаходитися в деякому діапазоні. Інший приклад обмеження: значення стовпця не повинно бути невизначеним.

У табл. 8 наведені основні обмеження для стовпців. Однак існують і більш складні обмеження.

*Таблиця 8.* Обмеження для стовпців

| **Визначення** | **Опис** |
| --- | --- |
| NOT NULL | Стовпець не може містити значення NULL, тобто значення цього стовпця повинні бути визначеними. |
| UNIQUE | Значення, введені в стовпці, мають відрізнятися від всіх інших значень у цьому стовпчику, тобто бути унікальними. |
| PRIMARY KEY | Стовпець є первинним ключем. У кожній таблиці тільки один стовпець може бути первинним ключем. Це означає, що він не може містити значення NULL, a введенні значення мають відрізнятися від всіх інших значень у цьому стовпчику. Таким чином, PRIMARY KEY є комбінацією NOT NULL і UNIQUE. |
| DEFAULT(*значення)* | Встановлює значення за замовчуванням. Так, при додаванні нового запису стовпець з таким обмеженням автоматично отримає вказане значення. |
| CHECK*(умова)* | Дозволяє проводити перевірку умови при введенні даних. Значення буде збережено, якщо умова виконується, у протилежному випадку – ні. |

Деякі зауваження про використання NULL (використання NOT NULL).

Від спроможності стовбця приймати значення null залежить можливість рядків містити NULL, тобто обов’язковість введення значень в цих рядках.

Властивості null:

􀂄 значення null є маркером того, що значення не було введене;

􀂄 значение null відображає значення, якого немає або невідоме або неприйнятне (але не значення нуль).

􀂄 значение null – не теж саме, що (0), пусте поле або пробіл (' ');

􀂄 значение null не відносить до жодного типу даних і може бути вставлене в будь-який стовбець, який це допускає;

􀂄 в командах SQL слово NULL вказує на значение null.

При встановленні обмеження на значення null потрібно враховувати наступне:

􀂄 обмеження на значення null завжди є обмеженням на стовбець, а не таблицю (про це далі);

􀂄 вы задаєе обмеження на значення null за допомогою ключевих слів NULL или NOT NULL в позначенні стовбця в команді CREATE TABLE;

􀂄 бажано уникати дозволу на значення null, оскільки це ускладнює запити;

􀂄 заборона значення null в стовбці може допомогти зберегти цілістність даних, оскільки це вимога на конкретне значення. СУБД не буде вставляти або змінювати рядок, якщо стовбець (для якого заборонене значение null) міститьNULL;

􀂄 деякі обмеження (наприклад, PRIMARY KEY) не можуть використовуватися в стовбцях, для яких дозволено значення null;

􀂄 значення null впливає на обмеження вторинних ключів (FOREIGN KEY);

􀂄 якщо ви додали рядок командою INSERT, не вказавши значення стовбця, який допускає значення null, ваша СУБД вставит NULL (при умові, що нема обмеження DEFAULT);

􀂄 ви можете ввести NULL безпосередньо в стовбці, для которого разрешено значение null, незалежно від того, який тип даних або замовчуване значення вказані для цього стовбця;

􀂄 якщо ви не вкажете NULL або NOT NULL, по замовчуванню значення null буде дозволено.

PRIMARY KEY

Головне про первинні ключі:

􀂄 первинний ключ ідентифікує кожний рядок в таблиці як уникальний;

􀂄 два рядка не можуть мати однаковий первиний ключ;

􀂄 первинні ключі не можуть містити NULL;

􀂄 в кожній таблиці може бути лише один первинний ключ;

􀂄 первинний ключ – це стовбець або декілька стовбців. *Простий первинний ключ* складається з одного стовбця; *складений* – з декількох;

􀂄 в *складеному ключі* значення в одном стовбці можуть повторюватися, але комбінація значень у всіх стовбцях ключа повинна бути унікальною;

􀂄 таблиця може мати декілька комбінацій стовбців, які ідентифікують її рядки.

При установці обмеження первинного ключа потрібно мати на увазі:

􀂄 простий ключ завжди може бути як обмеженням стовбця, так і обмеженням

таблиціскладений ключ може бути тільки обмеженням таблиці;

􀂄 якщо PRIMARY KEY є обмеженням таблиці, вам потрібно задати назву стовбця (стовбців). Якщо PRIMARY KEY є обмеженням столбца, воно відноситься до відповідному стовбцю;

􀂄 SQL дозволяє створювати таблицю без первинного ключа (порущуючи вимоги моделі таблиці). На практиці вам завжди потрібно задавати первинний ключ для будь-якої таблиці;

􀂄 для однієї таблиці дозволено задавати не більшше одного первинного ключа;

􀂄 обмеження первинних ключів маже завжди мають унікальні назви. Щоб задавати назву первинного ключа, користуйтеся виразом CONSTRAINT;

􀂄 обмеженням на значення null для стовбців первинних ключів повинно бути NOT NULL. Якщо вы не вкажете це обмеження, СУБД автоматично задасть його для всіх стовбців первинних;

􀂄 при додаванні рядків за допомогою команди INSERT потрібно вказати значення для первинного ключа, інакше вони будуть згенерировані автоматично у відповідності з типом даних стовбця.Значення первинного ключа рідко змінюються, не рекомендується змінювати значення первинних ключів командою UPDATE;

􀂄 не використовуйте для інших рядків значення первинного ключа, вилученого за допомогою команди DELETE рядка ;

**Приклади завдання простого первинного ключа**

**1.** Задати простий первинний ключ без назви для таблицы publishers за допомогою обмежень таблиці

CREATE TABLE publishers

(

pub\_id CHAR(3) NOT NULL,

pub\_name VARCHAR(20) NOT NULL,

city VARCHAR(15) NOT NULL,

state CHAR(2) NULL ,

country VARCHAR(15) NOT NULL,

**PRIMARY KEY (pub\_id)**

);

**2.** Задати простой первинний ключ для таблиці publishers за допомогою обмеження стовбців

CREATE TABLE publishers

(

pub\_id CHAR(3) **PRIMARY KEY**,

pub\_name VARCHAR(20) NOT NULL ,

city VARCHAR(15) NOT NULL ,

state CHAR(2) NULL ,

country VARCHAR(15) NOT NULL

);

**3.** Задати простий первиний ключ без назви для таблицы publishers за допомогою обмеження таблиці

CREATE TABLE publishers

(

pub\_id CHAR(3) NOT NULL,

pub\_name VARCHAR(20) NOT NULL,

city VARCHAR(15) NOT NULL,

state CHAR(2) NULL ,

country VARCHAR(15) NOT NULL,

**PRIMARY KEY (pub\_id)**

);

**4.** Задати іменований простий первиний ключ для таблицы publishers за допомогою обмеження таблиці

CREATE TABLE publishers

(

pub\_id CHAR(3) NOT NULL ,

pub\_name VARCHAR(20) NOT NULL ,

city VARCHAR(15) NOT NULL ,

state CHAR(2) NULL ,

country VARCHAR(15) NOT NULL ,

**CONSTRAINT publishers\_pk**

**PRIMARY KEY (pub\_id)**

);

**5.** Задати складений первинний ключ для таблиці title\_authors за допомогою обмеження таблиці

CREATE TABLE title\_authors

(

title\_id CHAR(3) NOT NULL,

au\_id CHAR(3) NOT NULL,

au\_order SMALLINT NOT NULL,

royalty\_share DECIMAL(5,2) NOT NULL,

**CONSTRAINT title\_authors\_pk**

**PRIMARY KEY (title\_id, au\_id)**

);

*Microsoft Access не підтримує ключові слова DEFAULT і CHECK у визначеннях обмежень.*

Розглянемо, як приклад, створення таблиці Titles (ISBN, Title, PubID, YearPub, Description, Notes, Subject, Comments). ISBN (стрічкове значення) повинен однозначно ідентифікувати запис про студента, тобто мати визначені і унікальні значення. Таким чином, даний стовпець повинен бути первинним ключем. Від стовпців Title, PubID та YearPub вимагатимемо, щоб у них не було невизначених значень. Поле Comments по замовчуванню повинне бути “Без коментарів”. Тоді запит на створення такої таблиці буде мати вигляд:

CREATE TABLE Titles (

ISBN CHAR(20) PRIMARY KEY,

Title VARCHAR NOT NULL,

PubID INTEGER NOT NULL,

YearPub DATE,

Description VARCHAR,

Notes VARCHAR,

Subject VARCHAR,

Comments VARCHAR DEFAULT(‘Без коментарів’)

);

Припустимо, що даний запит виконаний. Таблиця буде створена. Тоді, запит на додавання порожнього запису в таблицю Titles викличе повідомлення про помилку і не буде виконаний. Це відбудеться тому, що в прожньому записі, всі стовпці містять значення NULL, а, отже, не виконуються обмеження для перших чотирьох стовпців. При спробі добавити перший запис в таблицю Titles із значеннями (‘978-5-699-79339-6’, ‘Sherlock Holmes: ’, 4), буде виконаний і не викличе проблем з боку СКБД.

Якщо слідом за даним запитом ми спробуємо додати ще один запис зі значеннями (978-5-699-79339-6, ‘Unknown adventures of Sherlock Holmes’, 6). Даний запит не буде виконаний через порушення обмеження, накладеного на перший стовпець: його значення повинні бути не тільки певними (відмінними від NULL), але й унікальними. Запис можна додати, якщо стовпцю ISBN привласнити інше значення.

В наступному прикладі подано створення таблиці Authors, в якої поле AUID цілочисельне і є первинним ключем, а значення в полі Contracted не повинні перевищувати значення 1000.

CREATE TABLE Authors (

AUID INTEGER PRIMARY KEY,

Surname VARCHAR NOT NULL,

Name VARCHAR,

Lastname VARCHAR,

Address VARCHAR,

Contracted Numeric CHECK (Contracted<=1000)

);

У наступному прикладі створюється Таблиця Authors (Surname, Name, Lastname, Address, Contracted). Передбачається, що комбінація значень перших трьох стовпців повинна однозначно ідентифікувати запис у таблиці, тобто бути первинним ключем.

CREATE TABLE Authors (

Surname VARCHAR NOT NULL,

Name VARCHAR, NOT NULL

Lastname VARCHAR, NOT NULL

Address VARCHAR,

Contracted Numeric,

CONSTRAINT PRIMARY KEY (Surname ,Name, Lastname)

);

При добавленні наступних значень в таблицю:

(‘Tretyak’, ‘Alina’, ‘Petrivna’) та (‘Tretyak’, ‘Alina’, ‘Sergiivna’)

дані будуть добавлені, оскільки комбінації значень стовпців, що визначають первинний ключ, не містять NULL і відрізняються один від одного.

**Обмеження для таблиць**

Обмеження на дані, що вводяться можна призначити не тільки для окремих стовпців, але і для таблиці в цілому. Це зручно в тих випадках, коли необхідно для декількох стовпців призначити однакові обмеження. Крім того, якщо первинний ключ складений (тобто складається з декількох стовпців), то вказати його можна тільки як обмеження для таблиці, а не для стовпця. Всі обмеження для таблиці вказуються після визначень стовпців і складаються з ключового слова CONSTRAINT, за яким слідує вираз, що визначає саме обмеження. У табл. 9 наведені основні обмеження для таблиці.

*Таблиця 9.* Обмеження для таблиці

|  |  |
| --- | --- |
| **Визначення** | **Опис** |
| UNIQUE(*Список Стовпців)* | Значення в стовпцях, зазначених у списку повинні бути унікальними. |
| PRIMARY KEY*(Список Стовпців)* | У кожній таблиці повинен бути тільки один первинний ключ, який визначений або як обмеження для стовпця, або як обмеження для таблиці. Якщо первинний ключ складний, то він повинен бути визначеним як обмеження для таблиці. Праворуч від ключових слів PRIMARY KEY в круглих дужках вказується список стовпців, що визначають складений первинний ключ. |
| CHECK*(умова)* | Дозволяє проводити перевірку умови при введенні даних. Значення буде збережено, якщо умова виконується, у протилежному випадку – ні. На відміну від обмеження для стовпця, тут можна використовувати умови, що оперують значеннями різних стовпців таблиці. |
| FOREIGN KEY … REEERENCES… | Обмеження типу |

У наступному прикладі створюється Таблиця Authors (Surname, Name, Lastname, Address, Contracted). Передбачається, що комбінація значень перших трьох стовпців повинна однозначно ідентифікувати запис у таблиці, тобто бути первинним ключем.

CREATE TABLE Authors (

Surname VARCHAR NOT NULL,

Name VARCHAR, NOT NULL

Lastname VARCHAR, NOT NULL

Address VARCHAR,

Contracted Numeric,

CONSTRAINT PRIMARY KEY (Surname ,Name, Lastname)

);

При добавленні наступних значень в таблицю:

(‘Tretyak’, ‘Alina’, ‘Petrivna’) та (‘Tretyak’, ‘Alina’, ‘Sergiivna’)

дані будуть добавлені, оскільки комбінації значень стовпців, що визначають первинний ключ, не містять NULL і відрізняються один від одного.

**Зовнішні ключі**

Одна з найважливіших різновидів обмежень пов’язана з визначенням зовнішніх ключів. Зовнішній ключ – це стовпець або група стовпців, відповідних первинному ключу іншої таблиці. Щоб зрозуміти синтаксис виразу, що визначає зовнішній ключ, розглянемо приклад.

Нехай в базі даних є дві таблиці (для прикладу візьмемо спрощені таблиці бази даних “Книжковий магазин”):

**Titles** (ISBN, Title, YearPub) – містить список назв книг;

**BookSales** (BookSalesID, Sales, ISBN, Units) відомості про продажі книг.

У таблиці Titles стовпець ISBN є первинним ключем, тобто його значення відмінні від NULL і унікальні. У таблиці BookSales стовпець ISBN не зобов’язаний мати унікальні значення, оскільки одна і та ж книга може продаватися в різний час і по різній ціні. Разом з тим будь-якому значенню стовпця BookSales.ISBN відповідає єдине значення стовпця Titles.ISBN. При описаних умовах стовпець ISBN таблиці BookSales є зовнішнім ключем, який посилається на первинний ключ ISBN таблиці Titles.

Зовнішній ключ визначається як обмеження для таблиці в виразі з ключовими словами CONSTRAIN FOREIGN KEY (обмеження “зовнішній ключ”):

CONSTRAINT FOREIGN KEY ЗовнішнійКлюч REFERENCES зовнішня Таблиця (ПервиннийКлюч)

Тут ЗовнішнійКлюч – ім’я стовпця або список стовпців, розділених комами, які визначають зовнішній ключ, за ключовим словом REFERENCES (посилання) вказується зовнішня таблиця і її первинний ключ, на який посилається зовнішній ключ.

Для розглянутого раніше прикладу таблицю BookSales можна визначити наступним чином:

CREATE TABLE BookSales (

BookSalesID INTEGER PRIMARY KEY,

Sales NUMERIC,

ISBN CHAR(20),

Units NUMERIC,

CONSTRAINT FOREIGN KEY ISBN REFERENCES Titles (ISBN)

);

Використання зовнішніх ключів забезпечує збереження посилкової цілісності бази даних при зміні і видаленні записів. Якби таблиці Titles і BookSales не були пов’язані, то при видаленні записів з таблиці Titles у таблиці BookSales могли залишитися посилання на книгу, про яку вже немає відомостей. Цей факт зазвичай розцінюється як аномалія видалення. У випадку визначення в таблиці BookSales зовнішнього ключа ISBN, з таблиці Titles не вдасться видалити книгу, якщо ця книга продається в магазині. Якщо потрібно видалити з бази даних все, що стосується певної книги, то спочатку видаляються записи в таблиці BookSales, а потім – з таблиці Titles.

Аналогічна ситуація може статися і при оновленні даних. Наприклад, у таблиці Titles ви змінили ідентифікатор книги, яка має відомості в таблиці BookSales і, таким чином, зовнішньому ключу тепер немає на що посилатися. Це аномалія зміни. В даному випадку необхідно спочатку додати новий запис в таблицю Titles, вказавши в ній необхідне значення ISBN, потім змінити в таблиці BookSales всі старі значення ISBN на ті, які ви тільки що ввели в новому записі таблиці Titles, а потім видалити з таблиці Titles запис зі старим ISBN.

Щоб в таблицях, пов’язаних зовнішнім ключем, не робити модифікацію даних в декілька етапів, у виразі CONSTRAIN FOREIGN KEY можна використовувати додаткові ключові слова:

- ON DELETE CASCADE | SET NULL (при видаленні каскадувати | встановити NULL);

- ON UPDATE CASCADE | SET NULL (при оновленні каскадувати | встановити NULL).

Тут вертикальна риска не є елементом синтаксису, а лише розділяє можливі варіанти ключових слів.

Так, при використанні ON DELETE CASCADE у випадку видалення запису зі значенням первинного ключа, яке є у зовнішньому ключі іншої таблиці, відповідні записи видаляються автоматично з двох таблиць.

Наприклад, при видаленні з таблиці Titles записи про книгу, що має дані про продажі, в таблиці BookSales також будуть видалені всі записи, що посилаються на дану книгу. Щоб дана стратегія виконувалася, таблиця BookSales повинна бути визначена таким чином:

CREATE TABLE BookSales (

BookSalesID INTEGER PRIMARY KEY,

Sales NUMERIC,

ISBN CHAR(20),

Units NUMERIC,

CONSTRAINT FOREIGN KEY ISBN REFERENCES Titles (ISBN) ON DELETE CASCADE);

Однак на практиці зазвичай стараються спочатку переконатися, що для книги немає відомостей про її продажі, і лише потім викреслити її з списку наявних книг.

Варіант SET NULL зазвичай використовується при оновленні даних. Наприклад:

CREATE TABLE BookSales (

BookSalesID INTEGER PRIMARY KEY,

Sales NUMERIC,

ISBN INTEGER,

Units NUMERIC,

CONSTRAINT FOREIGN KEY ISBN REFERENCES Titles (ISBN) ON UPDATE SET NULL);

В даному випадку при зміні (у тому числі і при видаленні) в таблиці Titles запису, на який посилається зовнішній ключ таблиці BookSales, значення зовнішнього ключа встановлюються в NULL. Проте цей варіант не спрацює, якщо на стовпець замовлення, ISBN накладено обмеження NOT NULL. Зазвичай так і буває, оскільки при продажі книги, книга повинна бути в списку наявних книг в магазині. Тому, на всяк випадок, краще використовувати ключові слова ON UPDATE CASCADE.

*Деякі СКБД (наприклад, PostgreSQL) допускають комбінування додаткових ключових слів. Наприклад:*

*ON UPDATE SET NULL ON DELETE CASCADE*

**Видалення таблиць**

Видалити таблицю з бази даних можна наступним чином:

DROP TABLE Ім’яТаблиці;

Зрозуміло, що при видаленні таблиці втрачаються і всі дані що містяться в ній. Під час роботи з базою даних нерідко створюється таблиця для тимчасового зберігання даних, отриманих на якомусь проміжному етапі. Рано чи пізно такі таблиці підлягають видаленню. Однак можна забути це зробити. Крім того, проміжні таблиці, що створюються додатками, можуть залишитися в базі даних через збої. Тому для створення тимчасових таблиць краще використовувати оператор CREATE TEMPORARY TABLE, а не CREATE TABLE. Тимчасова таблиця, створена за допомогою CREATE TEMPORARY TABLE, автоматично знищується по закінченні сеансу роботи з базою даних.

**Модифікація таблиць**

При розробці бази даних рідко коли вдається відразу оптимально визначити структуру таблиць, що входять до неї. Крім того, вже готова база даних з часом може поповнюватися новими таблицями, які зв’язуються з уже наявними. А установка нових зв’язків вимагає корекції параметрів в старих таблицях. Модифікація таблиці – це зміна її структури, тобто додавання, видалення і перейменування стовпців, а також зміна їх типів і розмірів.

Мова SQL володіє спеціальними засобами модифікації таблиць.

Значно простіше змінити структуру таблиці за допомогою оператора ALTER TABLE (змінити таблицю). За допомогою додаткових ключових слів можна виконати наступні операції:

- ADD COLUMN – додати стовпець;

- DROP COLUMN – видалити стовпець;

- ALTER COLUMN – змінити тип, розмір і обмеження стовпця;

- RENAME COLUMN – перейменувати стовпець;

- RENAME ТО – перейменувати таблицю.

Типовою задачею зміни структури таблиці є додавання стовпця. Це можна зробити за допомогою SQL-виразу з таким синтаксисом:

ALTER TABLE Ім’яТаблиці ADD COLUMN ім’яСтовпця тип (розмір)

Наприклад: ALTER TABLE Authors ADD COLUMN Telephone CHAR (25);

Доданий стовпець виявляється останнім у таблиці, тобто займає крайню праву позицію. Іноді це незручно. Нехай, наприклад, стовпець Name в таблиці займає першу позицію, а раніше забутий і пізніше доданий стовпець Surname – сьому позицію. Щоб вибірка даних з цієї таблиці вивелася звичним чином, доводиться спеціально вказувати необхідне розташування стовпців в операторі SELECT, наприклад, SELECT Name, Surname FROM.

Вираз ADD COLUMN може використовуватися у виразі ALTER TABLE кілька разів – для кожного окремого стовпця. При визначенні параметрів доданого стовпця можна вказати обмеження для нього. Наступний SQL-вираз додає в таблицю Authors стовпець AUID і оголошує його первинним ключем (якщо при створенні первинний ключ не був вказаний):

ALTER TABLE Authors ADD COLUMN AUID INTEGER PRIMARY KEY;

He слід забувати, що первинний ключ у таблиці може бути тільки один. Якщо в таблиці вже є первинний ключ, то додавання стовпця як первинного ключа не буде виконано. Якщо вам потрібно перепризначити первинний ключ, то спочатку необхідно скоригувати відповідним чином параметри вже наявного стовпця, оголошеного як первинний ключ, а потім додати новий стовпець з параметром PRIMARY KEY. Корекцію стовпця можна виконати так: або спочатку видалити його, а потім додати новий стовпець з необхідними параметрами або змінити параметри існуючого стовпця за допомогою оператора ALTER COLUMN.

Щоб видалити стовпець з таблиці, досить виконати SQL-вираз з наступним синтаксисом:

ALTER TABLE Ім’яТаблиці DROP COLUMN ім’яСтовпця;

При видаленні стовпця видаляються всі дані, що містяться в ньому. Якщо видаляється стовпець який є первинним ключем, на який посилається зовнішній ключ з іншої таблиці, то видалення не буде виконано. У цьому випадку вам доведеться спочатку скоригувати дані в іншій таблиці.

Наприклад, наступний вираз видаляє з таблиці Authors стовпець Contracted.

ALTER TABLE Authors DROP COLUMN Contracted;

Для зміни параметрів існуючого стовпця, таких як тип, розмір і обмеження, застосовуються ключові слова ALTER COLUMN:

ALTER TABLE Ім’яТаблиці ALTER COLUMN ім’яСтовпця тип (розмір) [Обмеження];

Тут квадратні дужки вказують, що укладений в них вираз не є обов’язковим.

У наступному прикладі в таблиці Authors вже існуючий стовпець Name набуває нові параметри. А саме, будучи символьним, він отримує збільшення довжини до 50 символів і стає первинним ключем:

ALTER TABLE Authors ALTER COLUMN Name CHAR (50) PRIMARY KEY;

Зрозуміло, якщо в таблиці вже є первинний ключ (наприклад, стовпець AUID), то зробити первинним ключем ще один стовпець не вдасться. Крім того, не слід забувати, що при перетворенні типів можуть бути втрачені дані. Так, якщо ви перетворите стовпець символьного типу в числовий тип, то всі дані будуть втрачені. При зменшенні розміру символьного стовпця його значення можуть виявитися обрізаними праворуч. Таким чином, слід дуже уважно змінювати параметри стовпців, що містять деякі дані.

Перейменувати стовпці і таблицю можна за допомогою оператора ALTER TABLE З ключовими словами RENAME COLUMN ... ТО (перейменувати стовпець ... в).

Для перейменування стовпця використовується наступний синтаксис:

ALTER TABLE Ім’яТаблиці RENAME COLUMN ім’яСтовпця ТО нове\_ім’яСтовпця;

При перейменуванні стовпця його тип, розмір і обмеження зберігаються.

Для перейменування таблиці використовується схожий синтаксис:

ALTER TABLE Ім’яТаблиці RENAME TO нове\_Ім’яТаблиці;

Зміна структури таблиці є небезпечною операцією з точки зору можливості втрати даних. Якщо ви не цілком упевнені, що отримаєте потрібний результат, то краще використовувати резервне копіювання даних, наступним чином:

1. Створіть тимчасову таблицю за допомогою оператора CREATE TEMPORARY TABLE.

2. Скопіюйте в тимчасову таблицю дані з таблиці, структуру якої ви збираєтеся модифікувати.

3. Далі можна зробити двома способами:

- змінити структуру вихідної таблиці так, як вам треба. Цей спосіб краще застосовувати при корекції структури, не пов’язаної зі змінами існуючих обмежень на стовпці і / або на таблицю в цілому;

- видалити вихідну таблицю і створити нову під тим же ім’ям і з необхідною структурою.

4. Скопіюйте дані з тимчасової таблиці в таблицю з новою структурою, використовуючи оператор INSERT INTO. Більше тимчасова таблиця не потрібна, вона буде автоматично видалена після закінчення сеансу роботи з базою даних.

**SELECT INTO**

Формат команди

SELECT *columns* INTO *new\_table* FROM *existing table*

[WHERE *search\_condition*];

*new\_table* – це назва нової таблиці, яку створюють;

*existing\_table* – назва таблиці-джерела;

*columns* – список розділених комами виразів або назв стовбців з *existing\_table*. СУБД розраховує вираз в *columns* , щоб визначити структуру *new\_table*. Стовбці в *new\_table* створюються в порядку, заданому *columns*. Кожний стовбець в *new\_table* має таку ж назву, тип даних та значення, що і відповідний стовбець в *columns*. Вираз WHERE визначає, які рядки з *existing\_table* будуть додані в *new\_table*. Можете вказати вирази GROUP BY, HAVING та JOIN. *search\_condition* – це будь-яка умова пошуку WHERE

Команда SELECT INTO створює нову таблицю та заповнює її результатами виконання команди SELECT. Ця команда аналогічна створенню пустої таблиці за допомогою CREATE TABLE та заповненню її за допомогою INSERT SELECT (буде розглянуте при вивченні команди INSERT). Команда SELECT INTO експортує рядки з існуючої таблиці, а команда INSERT SELECT імпортує рядки в існуючу таблицю. Зазвичай команда SELECT INTO використовується для:

􀂄 архивування окремих рядків;

􀂄 створення резервних копій даних;

􀂄 створення копії таблиці у визначений момент часу;

􀂄 швидкого копіювання структури таблиц без її даних;

􀂄 створення даних для тестуванняя;

􀂄 копіювання таблиці для безпечної перевірки команд INSERT, UPDATE та DELETE.

При використання команди SELECT INTO потрібно враховувати:

􀂄 вы можете вибирати рядки для нової таблиці за допомогою стандартних виразів WHERE, JOIN, GROUP BY и HAVING команди SELECT або за допомогою будь-якої опції SELECT (SELECT розглядається пізніше);

􀂄 ця команда додає рядки в одну таблицю, незалежно від того, скільки таблиц-джерел використовує команда SELECT; властивості стовбців та виразів в списку втразів команди SELECT визначає структуру нової таблиці;

􀂄 якщо ви включите в список пунктів команди SELECT стовбець з довільними (розрахованими) даними, то значення в стовбцях нової таблиці будуть відповідати значенням, які були розраховані на момент виконання команди SELECT INTO;

􀂄 назва нової таблиці повинна різнитися від назви існуючої;

􀂄 для використання цієї команди, потрібно мати права на виконання команди CREATE TABLE, надані администратором вашої БД.

**Приклади**

**1.** Копіювати структуру та дані існуючої таблиці authors в новую таблицю authors2

**SELECT** \*

**INTO** authors2

FROM authors;

**2.** Копіювати тільки структуру (але не дані) існуючої таблиці publishers в нову таблицу с названим publishers2

**SELECT** \*

**INTO** publishers2

FROM publishers

WHERE 1 = 2;