Лабораторна робота №6
Тема: Генератори. Тригери. Конструкції мови SQL.

Теоретичні відомості:

Генератор – це механізм, який створює послідовний унікальний номер, який автоматично вставляється в стовпець під час таких операцій, як **INSERT** або **UPDATE**. Генератори зазвичай використовуються для створення унікальних значень, які можуть бути вставлені в стовпець, який використовується як первинний ключ.

1. Створення генератора

CREATE GENERATOR *ім'я_генератора*;

При створенні генератора за замовчуваннчм його початкове значення дорівнює нулю.

2. Ініціалізація генератора

SET GENERATOR *ім'я_генератора* **TO** *ціле_число***;**

3. Функція GEN_ID

GEN_ID (*ім'я_генератора* **,** *ціле_число***);**

Дана функція збільшує поточне значення генератора на ціле число.

Приклад:

Створити генератор, який використовується для створення унікальних значень поля SNOM, який є первинним ключем таблиці, починаючи з 101.

```
CREATE GENERATOR GEN_STUDENTS;
SET GENERATOR GEN_STUDENTS TO 100;
INSERT INTO STUDENTS (SNOM, SFAM)
VALUES(GEN_ID(GEN_STUDENTS,1), 'IВАНОВ');
```

Тригери – це частина програмного коду, що асоціюється з таблицею або виглядом, яка автоматично виконує дії при додаванні, зміні або видаленні запису в таблиці або вигляді.

4. Створення тригера

```
CREATE TRIGGER ім'я_тригера FOR ім'я_таблиці
[ACTIVE | INACTIVE]
{BEFORE | AFTER}
{DELETE | INSERT | UPDATE}
[POSITION номер]
AS тіло_тригера
термінатор
```

[ACTIVE | INACTIVE] – необов'язковий параметр, який визначає дію тригера після завершення транзакції. **ACTIVE –** тригер використовується (по замовчуванню). **INACTIVE –** тригер не використовується.

{BEFORE | AFTER} – обов'язковий параметр, який визначає коли відбудеться активізація тригера до події (**BEFORE**) чи після (**AFTER**).

{DELETE | INSERT | UPDATE} – визначає операцію над таблицею, яка викликає тригер для виконання.

[POSITION *номер***] –** визначає порядок виклику тригера для обробки однієї події, наприклад при додаванні запису в таблицю. Номер має бути цілим від 0 до 32 767, по замовчуванню дорівнює нулю. Тригер, який має менший номер, виконується раніше.

тіло_тригера – складається з двох частин:

1) блок опису локальних змінних, які використовуються в тригері. Кожна змінна описується конструкцією

DECLARE VARIABLE *ім'я_змінної тип_змінної* ;

і закінчується крапкою з комою (;);

2) блок програмного коду, який починається оператором **BEGIN** і завершується оператором **END**

BEGIN
команди_InterBase
END

Кожна команда завершується крапкою з комою (;).

У блоці програмного коду тригера використовуються дві контекстні змінні: **OLD** і **NEW**. Змінна **OLD**.*ім'я_стовпця* відповідає за старі значення стовпця, змінна **NEW**.*ім'я_стовпця* – за нові значення.

Оскільки, символ крапка з комою, який використовується зазвичай для завершення SQL-команди, використовується також і всередині тригера, то для завершення команди **CREATE TRIGGER** слугує певний символ – термінатор**.** Це може бути будь-який символ, який не використовується в даній команді, наприклад **^** або **!! .**

Термінатор визначається перед створенням тригера командою

SET TERM *термінатор* ;

Після команди **CREATE TRIGGER** дію крапки з комою потрібно відновити командою

SET TERM ; *термінатор*

Приклад:

Створити генератор **GEN_STUDENTS** і створити тригер **STUDENTS_BI**, який використовує даний генератор як лічильник для поля **SNOM** при додаванні записів у таблицю **STUDENTS**.

```
CREATE GENERATOR GEN_STUDENTS;
SET TERM ^ ;
CREATE TRIGGER STUDENTS_BI FOR STUDENTS
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  IF (NEW.SNOM IS NULL) THEN
    NEW.SNOM = GEN_ID(GEN_STUDENTS,1);
END ^
SET TERM ; ^
```

5. Модифікація тригера

ALTER TRIGGER *ім'я_тригера*
[ACTIVE | INACTIVE]
[{BEFORE | AFTER} {DELETE | INSERT | UPDATE}]
[POSITION *номер***]**

[AS *тіло_тригера* **]**
[*термінатор***]**

При модифікації можна змінити статус тригера, порядок і спосіб його виконання, внести зміни у тіло тригера. Термінатор вказується, якщо модифікується тіло тригера.

Приклад:

Змінити статус активного тригера *TR1* на пасивний.

ALTER TRIGGER TR1 **INACTIVE;**

6. Видалення тригера

DROP TRIGGER *ім'я_тригера***;**

Конструкції мови SQL

1. Коментар

/* *коментар* ***/**

2. Конкатенація

'симв_рядок1'||'симв_рядок2'

3. Оператор умови IF ... THEN ... ELSE ...

IF (*умова***)**
THEN *оператор_1*
[ELSE *оператор_2***];**

умова – логічний вираз, який має значення TRUE або FALSE;

оператор_1 – виконується, якщо умова приймає істинне значення. Замість *оператор_1* може бути блок операторів, обмежений конструкцією **BEGIN … END**;

оператор_2 – виконується, якщо умова хибна. Замість *оператор_2* може бути блок операторів, обмежений конструкцією **BEGIN … END**.

4. Оператор циклу WHILE … DO …

WHILE (*умова***) DO** *оператор;*

умова – логічний вираз, значення якого перевіряється перед кожним виконанням циклу;

оператор – оператор чи блок операторів **BEGIN … END**, який виконується, якщо умова приймає істинне значення.

5. Оператор SELECT

SELECT_команда INTO *список_змінних;*

Синтаксис даного оператора подібний до синтаксису команди **SELECT.** На відміну від звичайної команди **SELECT** результатом даної команди є один рядок, отримані дані записуються у змінні, вказані після ключового слова **INTO.**

6. Оператор циклу FOR … DO …

FOR SELECT_команда INTO *список_змінних* **DO** *оператор;*

SELECT_команда отримує дані з бази даних. Дана команда отримує за один

раз тільки один рядок, і результат записується у відповідні змінні, вказані після ключового слова **INTO.** Перед кожною змінною після **INTO** ставиться двокрапка (**:**) для того, щоб відрізнити ім'я стовпця від імені змінної;

оператор – оператор чи блок операторів **BEGIN … END**, який виконується для кожного рядка, отриманого командою **SELECT.**

Завдання до виконання:

1) Завантажте програму *IBExpert.*
2) Відкрийте базу даних **Univer**.
3) Створіть генератор **GEN1** з початковим значенням, рівним 20.
4) Створіть за допомогою тригера лічильник для поля **NOM** для додавання нових записів у таблицю **USPISH**.
5) Створіть тригер, який перетворює назви предметів таблиці **PREDMET** у прописні літери при модифікації чи додаванні нових записів у дану таблицю.
6) Створити тригер, який при знищенні запису з таблиці **STUDENTS** буде знищувати пов'язані з ним записи з таблиці **USPISH**.
7) Створити тригер, який при додаванні нових записів до таблиці **USPISH**, перевіряє чи є студент з відповідним номером в таблиці **STUDENTS**.
8) Створити тригер, який при видалені записів з таблиці **VUKLAD** заміняє значення зовнішнього ключа відповідних записів таблиці **PREDMET** на NULL.

Контрольні запитання:

1) Що таке генератор?
2) Якими командами задати створення і використання генератора?
3) Де може використовуватись функція **GEN_ID**?
4) Що таке тригер?
5) Які є види тригерів?
6) Чи може тригер використовуватись як окрема програма?
7) До яких дій по відношенню до таблиць приводить виконання тригерів?
8) Які оператори використовуються в тригерах?
9) Що означає слово POSITION 0 в описовій частині тригера?
10) Чи буде виконуватись тригер в описовій частині якого вказане слово INACTIVE?
11) Який зміст мають змінні NEW та OLD в тілі тригера?