Хязгаарлалт болон Триггерийн

хэрэглээ

(Лаборатори №2)

У. Төрболд

ХШУИС, Мэдээллийн технологийн IV-р түвшний оюутан, turbold1125@gmail.com

1. ОРШИЛ

Энэхүү лабораторын ажил нь SQL хэлийн хоёр чухал шинж чанар болох хязгаарлалт болон триггерийг судлах болно. Эдгээр функцууд нь өгөгдлийн бүрэн бүтэн байдлын дүрмийг дагаж мөрдөх, мэдээллийн сан дахь тодорхой үйл явдлууд дээр үндэслэн үйлдлүүдийг автоматжуулах боломжийг олгодог ба эдгээрийг ашиглан практик дээр туршина.

2. ЗОРИЛГО

Энэхүү лабораторын зорилго нь өгөгдлийн сан дээрх хязгаарлалт буюу constraint, триггер буюу автоматжуулалтыг хэрэгжүүлэх юм.

3. ОНОЛЫН СУДАЛГАА

3.1 Constraint

Тодорхойлолт:

Хүснэгтэнд орж болох өгөгдлийн төрлийг хязгаарлахад Constraint – ыг ашигладаг. Энэ нь хүснэгтэнд байгаа өгөгдлийн тодоорхой хязгаарлалт болон үнэн зөв, найдвартай байдлыг баталгаажуулдаг. Constraint болон өгөгдлийн үйлдлийн хооронд ямар нэгэн зөрчил байвал үйлдлийг зогсооно. Constraint нь баганын түвшин эсвэл хүснэгтийн түвшин байж болно. Баганын түвшний хязгаарлалт нь баганад, хүснэгтийн түвшний хязгаарлалт нь бүх хүснэгтэд хамаардаг.

Syntax:

```
CONSTRAINT constraint_name CHECK (column_expression)
```

- constraint_name: Шалгах хязгарлалтын хэрэглэгчийн тодорхойлсон нэр

- column_expression: Хүчинтэй утгыг Үнэн, буруу утгыг Худал гэж үнэлдэг логик илэрхийлэл

Давуу тал:

Өгөгдлийн баталгаажуулалтын үндсэн дүрмийг хэрэгжүүлэх энгийн бөгөөд үр дүнтэй арга.

Буруу оруулах, шинэчлэхээс сэргийлж мэдээллийн чанарыг сайжуулна.

3.2 Trigger

Тодорхойлолт:

SQL – д триггер нь датабааз дахь тодорхой хүснхэгт дээрх үйл явдлыг автоматаар ажилладаг хадгалагдсан процедур юм.

Syntax:

```
CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF} {INSERT | UPDATE | DELETE} ON table_name
[FOR EACH ROW]
[WHEN (condition)]
BEGIN
    -- Trigger (SQL statements)
END;
```

CREATE TRIGGER нь шинэ триггер үүсгэх эсвэл байгаа нэгийг солиход ашиглана.

BEFORE, AFTER нь хэзээ асах ёстойх заана.

INSERT, UPDATE, DELETE нь триггерийн үйлдлийн төрлийг заана.

WHEN нь нөхцөл

4. ХЭРЭГЖҮҮЛЭЛТ

4.1 Хүснэгт үүсгээд талбарт тус бүрт дараах хязгаарлалтыг хийж өг

1.1 Эхний орон ABCDEFGHKLMNP үсгүүдийн аль нэг нь байх, 2 дахь орон нь 1-9 хооронд тоо байх, сүүлийн орон нь ZYXLMNO үсгүүдийг агуулаагүй байна. 3 дахь оронгоос эхлээд сүүлийн оронгийн хооронд дурын урттай тоо, үсэг байж болох баганад зориулж CK бич
1.2. баганын утга нь 0 ээс их 80 аас бага байх шаардлагатай CK бич

1.3. эхний багана B25AP -аас өөр утгатай байх бөгөөд 2 дахь багана нь 42,53,65 утгыг агуулаагүй байх хүснэгтийн хувьд CK бич

1.4. SISI id -г зөв оруулах CK бич

```
CREATE TABLE Constraint1 (
    FirstColumn VARCHAR(12),
    SecondColumn INT,
    SISI_ID VARCHAR(13),
    CONSTRAINT CK_First CHECK (FirstColumn LIKE '[ABCDEFGHKLMNP][1-9]%'),
    CONSTRAINT CK_Second CHECK (SecondColumn > 0 AND SecondColumn < 80),
    CONSTRAINT CK_Third CHECK (FirstColumn NOT IN ('B25AP') AND SecondColumn NOT IN (42, 53, 65)),
    CONSTRAINT CK_SISI_ID CHECK (SISI_ID LIKE '[2][0-9][B][1][N][U][M][0-9][0-9][0-9][0-9]')
);
```

FirstColumn багана нь A-P хүртэлх утгыг аваад, 2 дахь орон нь 1-9 хүртэлх тоо байх ба энэхүү функцийг LIKE функцээр дамжуулан хэрэгжүүлсэн.

SecondColumn багана нь 0 – 80 хүртэл тоог жиших үйлдлээр хэрэгжүүлсэн.

CK_Third constraint нь FirstColumn B25AP утгыг агуулаагүй байх ба SecondColumn нь 42,53,65 гэх утгыг NOT IN функцийн тусламжтайгаар хэрэгжүүлсэн.

SISI_ID шалгахдаа эхний орон 2 – оор эхлэх ба дараагийн орон нь 0 – 9 хүртэлх цифр авах ба B гэсэн үсэг авч араас нь 1 гэсэн утгыг авна. 20B1NUM гэдэгтэй адилаар NUM гэх утгыг авч сүүлийн 4 орон нь цифр байх боломжтой.

Үр дүнг шалгах:

```
INSERT INTO Constraint1 (FirstColumn, SecondColumn, SISI_ID)
VALUES ('B25AP', 55, '20B1NUM4567');

INSERT INTO Constraint1 (FirstColumn, SecondColumn, SISI_ID)
VALUES ('B2A', 42, '20B1NUM4567');
```

Дээрх функцийг хэрэгжүүлэхэд алдаа гарах ба 42, B25AP гэсэн утгуудыг оруулж өгсөн нь CK_Third constraint – ийг зөрчиж байгаа тул доорх алдаа гарна.

```
Msg 547, Level 16, State 0, Line 17
The INSERT statement conflicted with the CHECK constraint "CK_Third". The conflict occurred in database "master", table "dbo.Constraint1".
The statement has been terminated.
```

Доорх функцийг хэрэгжүүлэхэд SISI_ID утга оруулахад NAM хэмээн оруулсан тул CK_SISI_ID constraint – ийг зөрчиж байгаа тул доорх алдаа гарна.

```
INSERT INTO Constraint1 (FirstColumn, SecondColumn, SISI_ID)
VALUES ('B2A', 50, '20B1NAM4567');
```

```
Messages
Msg 547, Level 16, State 0, Line 20
The INSERT statement conflicted with the CHECK constraint "CK_SISI_ID". The conflict occurred in database "master", table "dbo.Constraint1", column 'SISI_ID'.
The statement has been terminated.
```

Үр дүн:

Үүний дараа шалгаж дуусаад зөв утгуудыг оруулахад амжилттай болсон гэх мэдэгдлийг өгнө.

```
INSERT INTO Constraint1 (FirstColumn, SecondColumn, SISI_ID)
VALUES
    ('A1CK', 10, '20B1NUM4567'),
    ('B2CK', 20, '20B1NUM6543'),
    ('C3CK', 30, '20B1NUM7890'),
    ('D4CK', 40, '20B1NUM1234'),
    ('E5CK', 50, '20B1NUM5678'),
    ('F6CK', 60, '20B1NUM6789'),
    ('G7CK', 70, '20B1NUM8901'),
    ('H8CK', 79, '20B1NUM9012'),
    ('K9CK', 1, '20B1NUM1234');
```

```
Messages

(9 rows affected)

Completion time: 2024-03-08T20:21:32.5627476+08:00
```

```
SELECT * FROM Constraint1
```

Results | Messages

	FirstColumn	SecondColumn	SISI_ID
1	A1CK	10	20B1NUM4567
2	B2CK	20	20B1NUM6543
3	C3CK	30	20B1NUM7890
4	D4CK	40	20B1NUM1234
5	E5CK	50	20B1NUM5678
6	F6CK	60	20B1NUM6789
7	G7CK	70	20B1NUM8901
8	H8CK	79	20B1NUM9012
9	K9CK	1	20B1NUM1234

4.2 Өгөгдлийн санд өгөгдлийг програмын аргаар оруулах

2.1 Янз бүрийн өгөгдлийн төрөлтэй багануудтай хүснэгт үүсгэ. Int төрөлтэй 1 баганыг PK болгоно

```
CREATE TABLE ExampleTable (
    PK INT PRIMARY KEY,
    Column1 TEXT,
    Column2 INT
);
```

Энэ нь Text, Int төрөлтэй 3 багана байх ба PK баганыг Primary Key болгосон.

2.2 Хүснэгтэд мөр бичлэгийг нэмэхдээ PK баганын утгыг нэгээр нэмэгдүүлж бусад баганын мэдээллийг 10, 20, 30 яваад өөрчилөгдөхөөр програмчлана.

Жишээ нь:

1 texta 20
2 texta 20
...
10 textb 20
11 textb 20
...
20 textb 30
21 textb 30

Үүнийг програмчлахын тулд бидэнд нэг функц болон триггер ашиглан хийх боломжтой.

```
CREATE OR REPLACE FUNCTION generate_row_values()
RETURNS TRIGGER AS $$
BEGIN
    -- PK утгыг нэгээр нэмэгдүүлэх
    SELECT COUNT(*) INTO NEW.PK FROM ExampleTable;

    -- Хэрвээ ямар ч мөр нэмэгдээгүй тохиолдолд PK - ыг 1ээс эхлүүлэх
    IF NEW.PK = 1 THEN
    NEW.PK := 1;
    ELSE
    -- PK - ыг мөрийн тооноос хамаарч нэмэгдүүлэх
    NEW.PK := NEW.PK + 1;
    END IF;

    -- Column1, Column2 - ын утгыг PK - с хамааран тооцоолох

    -- 'a' - 'z' хүртэл 10 мөр тутамд нэмэгддэг байх
    NEW.Column1 := CHR(ASCII('a') + ((NEW.PK - 1) / 10) % 26);

    -- 10 мөр тутамд утга нь 10 - аар нэмэгддэг байх
    NEW.Column2 := 10 * ((NEW.PK - 1) / 10 + 1);

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Энэхүү функц нь ExampleTable хүснэгтэд шинэ мөр оруулахаас өмнө гүйцэтгэх зориулалттай.

COUNT(*) INTO NEW.PK нь тухайн хүснэгтийн мөрийг тоолж шинээр оруулсан мөрийн PK – д тоолох утгыг онооно.

IF NEW.PK = 1 THEN … END IF нь NEW.PK утгыг 1 – тэй тэнцүү эсэхийг шалгадаг. Хэрэв байгаа бол хүснэгтэд одоо мөр байхгүй гэсэн үг. NEW.PK утгыг 1 гэж оноосон ба мөр байгаа тохиолдолд NEW.PK утгыг 1 ээр нэмэгдүүлнэ.

1, 2 баганын утгыг тооцоолохдоо % ашиглан 10 мөр тутамд a – z, 10 – аар нэмэгддэг.

Return NEW нь өөрчилсөн шинэ мөрийн утгыг буцаана. Триггер функцийг ажиллуулсны дараа шинэчлэгдсэн утгууд бүхий мөрийн ExampleTable хүснэгтэнд оруулах болно.

```
CREATE TRIGGER SetRowValuesOnInsert
BEFORE INSERT ON ExampleTable
FOR EACH ROW
EXECUTE FUNCTION generate_row_values();
```

Энэхүү триггер нь ExampleTable – д утга оруулахын өмнө ажиллах ёстой ба хүснэгтэд оруулсан мөр бүрт триггер нэг удаа ажиллахыг тодорхойлсон ба дээрх функцийг гүйцэтгэнэ.

Үр дүн:

```
INSERT INTO ExampleTable (PK, Column1, Column2)
VALUES
    (1, 'a', 10),
    (10, 'b', 20),
    (20, 'c', 30),
    (30, 'd', 40),
    (40, 'e', 50),
    (50, 'f', 60),
    (60, 'g', 70),
    (70, 'h', 80),
    (80, 'i', 90),
    (90, 'j', 100),
    (100, 'k', 500),
    (50, 'l', 1000);
```

Data Output | Messages | Notifications

```
INSERT 0 12
```

	pk [PK] integer	column1 text	column2 integer
3	3	a	10
4	4	a	10
5	5	a	10
6	6	a	10
7	7	a	10
8	8	a	10
9	9	a	10
10	10	a	10
11	11	b	20
12	12	b	20
13	13	b	20
14	14	b	20
15	15	b	20
16	16	b	20
17	17	b	20
18	18	b	20
19	19	b	20
20	20	b	20
21	21	c	30
22	22	c	30
23	23	c	30
24	24	c	30

1.3 PK болсон багана дахь кодыг дараалсан биш байдлаар үүсгэдэг болгох

Жишээ нь

1
2
4

```
CREATE OR REPLACE FUNCTION generate_row_valuees()
RETURNS TRIGGER AS $$
DECLARE
    next_pk INT;
BEGIN
    -- next_pk олох
    SELECT COALESCE(MAX(PK) + 1, 1) INTO next_pk FROM ExampleTable;

    -- next_pk нь 3-т хуваагддаг бол 1-ээр нэмэгдүүлэн дараалсан бус болгоно.
    IF next_pk % 3 = 0 THEN
        next_pk := next_pk + 1;
    END IF;

    -- NEW_PK - д next_pk оноох
    NEW.PK := next_pk;

    -- Column1, Column2 - ын утгыг PK - с хамааран тооцоолох
    NEW.Column1 := CHR(ASCII('a') + ((NEW.PK - 1) / 10) % 26);
    NEW.Column2 := 10 * ((NEW.PK - 1) / 10 + 1);

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Pk баганаас хамгийн их утгыг сонгоод 1 ээр нэмэгдүүлэх замаар дараагийн боломжтой үндсэн түлхүүрийн утгыг олно. Хэрэв мөр байхгүй тохиолдолд 1ээс эхэлнэ. Next_pk нь 3-т хуваагддаг бол үүнийг дараалалгүй болгохын тулд 1 ээр нэмэгдүүлнэ. New.pk – ийн утгыг next_pk болгож өөрчилнө. Үүний дараа Trigger ийг Before Insert төлөвт ажиллуулдаг болгосноор доорх үр дүн гарна.

Үр дүн:

```
INSERT INTO ExampleTable (PK, Column1, Column2)
VALUES
    (1, 'a', 10),
    (10, 'b', 20),
    (20, 'c', 30),
    (30, 'd', 40),
    (40, 'e', 50),
    (50, 'f', 60),
    (60, 'g', 70),
    (70, 'h', 80),
    (80, 'i', 90),
    (90, 'j', 100),
    (100, 'k', 500),
    (50, 'l', 1000);
```

Data Output | Messages | Notifications

```
INSERT 0 12
```

	pk [PK] integer	column1 text	column2 integer
1	1	a	10
2	2	a	10
3	4	a	10
4	5	a	10
5	7	a	10
6	8	a	10
7	10	a	10
8	11	b	20
9	13	b	20
10	14	b	20
11	16	b	20
12	17	b	20

5. ДҮГНЭЛТ

Энэхүү лабораторын ажлаар SQL өгөгдлийн сан дахь хязгаарлалт, триггерүүдийн практик хэрэгжилтийг судалсан болно. Өгөгдөл оруулах дүрмийг тодорхойлох замаар өгөгдлийн бүрэн бүтэн байдал, үнэн зөв байдлыг хангахын тулд хязгаарлалтуудыг ашигласан бол өгөгдлийн сан дахь тодорхой үйл явдлуудад тулгуурлан үйлдлүүдийг автоматжуулахын тулд триггерийг ашигласан.