Лабораторийн ажил 9. ӨСБоловсруулалт-Триггер

Лабораторийн ажлын зорилго

- Триггер үүсгэх (Хугацаа, Event-г тохируулах)
- NEW, OLD объектийг ашиглах
- Триггер дотор T-SQL -ийн команд ашиглах
- Триггер ашиглах
- SINGAL буюу алдаа баригч ашиглах

Даалгавар-1

Оюутан бүр Бие даалтын ажлаар сонгосон сэдвийн дагуу байгуулсан ӨС схемийг ашиглаж ӨС үүсгэх скрипыг ажиллуулж , ӨС-г шинээр үүсгэх.

- Бие Даалт №1,2 дээр үүсгэсэн (Лавлах хүснэгтүүдэд бүх бичлэгийг оруулсан байх - Бодит утгууд оруулна) скрипт файлыг хуулж Лаборатори №9 файл үүсгэ
- Лаборатори №9 скрипт файлыг ажиллуулж ӨС-г үүсгэ

Даалгавар-2

Баг бүр Бие даалтын ажлаар сонгосон сэдвийн дагуу байгуулсан Функциональ шаардагад хэрэглэгдэх тооцооолох/боловсруулах үйлдлийг хийх Триггер үүсгэх.

- NEW, OLD объектийг ашиглах
- SINGAL буюу алдаа баригч ашиглах
- Шаардлагтай тооцооллыг хийж үр дүнг хадгалах
- Баг тус бүр дор хаяж ХОЁР ширхэг ОБЪЕКТЫН Триггер бүтээж БД-ыг тооцно

Хугацаа

Лабораторийн цаг дээр үзүүлнэ.

Гүйцэтгэж хамгаалах шаардлага

- Лабораторийн ажил №9 скрипт файлаас ӨС-ийн үүсгэх
- Даалгавар №2,3 -оор Триггер бүтээсэн кверийг тайлбарлах
- Бүтээсэн Триггерийг ажиллуулж, тайлбарлах

Нэмэлт материал

Онолын хэсэг, жишээ

Триггер

- SQL триггер гэдэг нь ӨС-ийн хүснэгтэнд ямар нэгэн event болох үед идэвхиждэг нэг
- болон хэд хэдэн SQL илэрхийлэл юм.
- Өөрөөр, тусгай зориулалтын процедур ч гэж ойлгож болно.
- Event нь нэмэх, устгах, засварлах гэх мэт үйлдлүүд байна.

Триггер үүсгэх

Синтакс:

```
CREATE TRIGGER триггер_нэр
      триггер_хугацаа триггер_event
      ON хүснэгт_нэр
   FOR EACH ROW триггер_код;
```

Триггерийн хугацаа ба Event

триггер_хугацаа триггер_event нь дараах утгуудыг авна:

- BEFORE INSERT: хүснэгтэнд өгөгдөл нэмэхийн өмнө.
- AFTER INSERT: хүснэгтэнд өгөгдөл нэмсэний дараа.
- BEFORE UPDATE: хүснэгтийн өгөгдлийг засварлахын өмнө.
- AFTER UPDATE: хүснэгтийн өгөгдлийг засварласаны дараа
- BEFORE DELETE: хүснэгтээс өгөгдөл устгахын өмнө
- AFTER DELETE: хүснэгтээс өгөгдөл устгасаны дараа

Нэг хүснэгтэнд ижил хугацаа болон event бүхий триггер байж болохгүй.
Жишээ нь:
Нэг хүснэгтэнд 2 BEFORE INSERT триггер байж болохгүй.
Харин нэг хүснэгтэнд BEFORE INSERT болон BEFORE UPDATE триггерүүд байж болно.

Хүснэгтийн нэр

Тухайн триггер ажиллах хүснэгт
Нэг триггер нь нэг л хүснэгттэй холбогдоно
View дээр триггер тодорхойлох боломжгүй.
Триггер_нэр:
Нэрлэхдээ нэрлэх дүрмээ баримтална: time_table_event

Жишээ нь: before_book_insert

Триггерийн боловсруулах/тооцоолох хэсэг

FOR EACH ROW:
Бичлэг нэмэх,засварлах, устгах үйлдэл бүрд триггер нь триггерийн кодонд бичигдсэн илэрхийлэл бичлэг тус бүр дээр ажиллана.
Триггер_илэрхийлэл:
Ажиллах илэрхийлэл
Нэгээс олон илэрхийлэл ажиллуулах бол: BEGIN/END блокыг ашиглах ба сторед процедуртай адил DELIMITER-г өөрчилнө.

Триггер

- SQL синтакс:

```
DELIMITER //
CREATE TRIGGER триггер_нэр
    тригге_хугацаа триггер_event
    ON хүснэгт_нэр
    FOR EACH ROW
    BEGIN
        триггер_код;
    END//
DELIMITER ;
```

Хэзээ, Ямар үйлдэлд

Ямар хүснэгтийн

Өөрчлөлт орсон мөр бүр дээр боловсруулалт хийх хэсэг

Триггер- Шинэ,хуучин мөрийг хадгалах объект

OLD түлхүүр үгээр өөрчлөлт хийгдэхээс өмнөх нэг мөр өгөгдлүүд
NEW түлхүүр үгээр шинээр нэмэгдсэн, засварлагдсан нэг мөр өгөгдлүүд хадгалагдана.

Ашиглагдах хэлбэр:
OLD.баганы_нэр: устгах, засварлагдахаас өмнөх утга
NEW.баганы_нэр: шинээр нэмэгдсэн утга, эсвэл засварлагдсан утга.

Триггер-Шинэ, хуучин бичлэг

	OLD	NEW
INSERT		+
UPDATE	+	+
DELETE	+	

Жишээ

Book хүснэгт:

	Field	Type	Null	Key	Default	Extra
▶	bok_id	int unsigned	NO	PRI	NULL	auto_increment
	bok_name	varchar(255)	YES		NULL	
	bok_price	decimal(9,2)	YES		NULL	
	buy_qty	int	YES		NULL	
	bok_status	varchar(25)	YES		NULL	

bok_status багананд шинээр бичлэг нэмэгдэх үед шинээр нэмэгдсэн гэсэн төлвийг хадгална.

Триггер: Жишээ

```
DELIMITER //
CREATE TRIGGER before_book_insert
BEFORE INSERT ON book
FOR EACH ROW
BEGIN
     SET NEW.bok_status = 'NEW';
END//
DELIMITER ;
```

UPDATE илэрхийллийн оронд
SET утга олгох илэрхийллийг ашиглана.

Триггерийг устгах

Синтакс:

```
DROP TRIGGER table_name.trigger_name;
```

Жишээ

```
DROP TRIGGER book.before_book_update;
```

Жишээ: Бичлэг нэмэх

Дараах командыг ажиллуулъя:

```
insert into book(bok_name, bok_price, buy_qty)
                                  values('Twillight', 100,1);
insert into book(bok_name, bok_price, buy_qty)
                               values ('Davinchi Code',10,2);
```

Нэмэх үйлдлийн дараа хүснэгтэнд дараах бичлэг нэмэгдэнэ

	id	bok_name	bok_price	buy_qty	bok_status
▶	1	Twillight	100.00	1	NEW
	2	Davinchi Code	10.00	2	NEW
	NULL	NULL	NULL	NULL	NULL

SIGNAL-Алдаа баригч

- SIGNAL алдаа барих, алдааны мэдээлэл тодорхойлоход ашиглагдана(SQLSTATE)
- Оруулах өгөгдөлд нөхцөл тавихад ашиглана.
- Ямар нэг утгыг шалгаад алдаатай буюу ямар нэг нөхцөлийг хангахгүй бол SIGNAL ашиглан MySQL-н алдааны мессежүүдийг гаргаж, нэмэх болон засварлах үйлдлүүдийг зогсоож болдог.
- Жишээ нь: sqlstate 45000: 'Unhandled user-defined exception condition'.

Жишээ

```
DELIMITER //
CREATE TRIGGER check_my_constrain BEFORE INSERT ON buy
    FOR EACH ROW
    BEGIN
    DECLARE msg varchar(255);
    IF (
      NEW.buy_qty <= 0
      OR NEW.buy_qty IS NULL
      OR NEW.buy_unit_price<(SELECT bok_price*0.9 FROM book WHERE bok_id=NEW.buy_bok_id))
    THEN
       SET msg = CONCAT('Алдаатай утга: тоо хэмжээ нь 0 болон NULL байж болохгүй' , NEW.buy_qty,
                  ' мөн unit_price нь үндсэн үнийн 90% -аас их байх ёстой', NEW.buy_unit_price);
      SIGNAL sqlstate '45000' SET message_text = msg;
    END IF;
    END //
DELIMITER ;
```

Buy хүснэгтэнд бичлэг нэмэх үед ажиллана :
Тоо хэмжээ утга нь зөв бол (NULL биш, 0-ээс их)
Нэгжийн үнэ нь үндсэн үнийн 90% байна

Алдаатай өгөгдөл байвал
алдааны мэдээлэл үүсгэнэ

Жишээ: Бичлэг нэмэх

Book хүснэгт:

bok_id	bok_name	bok_price	buy_qty	bok_status
1	Twillight	100.00	1	NEW
2	Davinchi Code-Edited	10.00	2	Updated
NULL	NULL	NULL	NULL	NULL

```
INSERT INTO buy (buy_ord_id,buy_bok_id, buy_qty,
                    buy_unit_price) VALUES (1,1,1,90);
```

→ Дээрх үйлдэл алдаагүй ажиллана

```
INSERT INTO buy (buy_ord_id,buy_bok_id,
buy_qty,buy_unit_price) VALUES (1,1,0,90);
```

→ нэмэх үйлдэл ажиллахгүй(qty=0)

Error Code: 1644. Алдаатай утга: тоо хэмжээ нь 0 болон NULL байж болохгүй:0 , мөн unit_price нь үндсэн үнийн 90% -аас их байх ёстой:90

Жишээ

→ Дараах үйлдэл

```
97  INSERT INTO buy (buy_ord_id,buy_bok_id, buy_qty, buy_unit_price) VALUES (1,1,1,85);
98  INSERT INTO buy (buy_ord_id,buy_bok_id, buy_qty, buy_unit_price) VALUES (1,1,0,100);
99  INSERT INTO buy (buy_ord_id,buy_bok_id, buy_qty, buy_unit_price) VALUES (1,1,-1,100);
100 INSERT INTO buy (buy_ord_id,buy_bok_id, buy_qty, buy_unit_price) VALUES (1,1,1,100);
101 select * from buy;
```

Action Output

#	Time	Action	Message
143	02:14:19	drop trigger if exists check_my_constraint	0 row(s) affected
144	02:14:19	CREATE TRIGGER check_my_constraint BEFORE INSERT ON buy FOR EACH ROW BEGIN DECLARE msg varchar(255); IF (...	0 row(s) affected
145	02:14:25	INSERT INTO buy (buy_ord_id,buy_bok_id, buy_qty, buy_unit_price) VALUES (1,1,1,85)	Error Code: 1644. Алдаатай утга: тоо хэмжээ нь 0 болон NULL байж болохгүй:1 мөн unit_
146	02:15:38	INSERT INTO buy (buy_ord_id,buy_bok_id, buy_qty, buy_unit_price) VALUES (1,1,0,100)	Error Code: 1644. Алдаатай утга: тоо хэмжээ нь 0 болон NULL байж болохгүй:0 мөн unit_
147	02:16:07	INSERT INTO buy (buy_ord_id,buy_bok_id, buy_qty, buy_unit_price) VALUES (1,1,-1,100)	Error Code: 1644. Алдаатай утга: тоо хэмжээ нь 0 болон NULL байж болохгүй:-1 мөн unit_
148	02:16:21	INSERT INTO buy (buy_ord_id,buy_bok_id, buy_qty, buy_unit_price) VALUES (1,1,1,100)	1 row(s) affected