sakila : 영화 dvd 관리하는 db

TABLE

--기본정보--

actor 배우 정보

address staff, customer, store 입력시 적은 address??

customer 손님 정보

film 영화 정보

store 매장 정보 -> 매장은 총 2곳

staff 직원 정보 -> 직원은 총 2명 -> 한 store에 staff 하나

--기본정보;분류--

category dvd 카테고리

city 각 도시 in country

country 각 나라별 코드

language film 상영 언어

--복합--

film_actor film-actor

film_category film-category

payment 렌탈 비용; 누가(customer) 어디서(store) 뭘(rental) 빌려가서 얼마가 나왔는 지?

rental dvd렌탈 내역; 누가(customer) 언제 어디서 무엇을 빌려갔는 지?

film_text film의 제목과 줄거리만 간단 요약

inventory 각 store에 dvd 몇 개 보유했는 지?

(+@_update : DATETIME)

(+last_update : TIMESTAMP; 저장되는 정보는 utc 기반?)

1. 트리거란?

특정 DB테이블의 동작(&이벤트; INSERT, UPDATE, DELETE)을 감시하고 있다가 정해놓은 조건에 해당하는 동작이 수행되는 순간 실행되는 기술(?or 프로그램)

2. 트리거 속성&키워드

트리거의 작동시점

-> before : 이벤트 발생 이전에 트리거 실행

-> after : 이벤트 발생 이후에 트리거 실행

동작(이벤트)

-> insert : 행을 삽입했을 때 트리거 실행

-> update : 행을 수정했을 때 트리거 실행

-> delete : 행을 삭제했을 때 트리거 실행

old : 예전 데이터

-> update : 수정 전의 데이터

-> delete : 삭제 전의 데이터

(-> insert는 기존의 데이터를 가지고 있지 않음)

new : 새 데이터

-> insert : 삽입 후의 데이터

-> update : 수정 후의 데이터

(-> delete는 데이터를 삭제하기 때문에 new를 가질 수 없음)

	old	new
Update	o	o
Insert	x	o
delete	o	x

3. 트리거 생성&삭제

트리거 생성

```
DELIMITER $$          //생략가능
CREATE TRIGGER 트리거명
        AFTER 이벤트 ON 기준테이블
        FOR EACH ROW      //실행될 문장 행에 각각 적용
BEGIN
        실행 쿼리
END $$
DELIMITER;            //생략가능
```

트리거 삭제

```
DROP TRIGGER 트리거명;
```

4. sakila의 트리거 살펴보기

-> ?? 왜 쿼리에 직접 now()를 넣지 않고, 트리거를 사용할까?

****customer_create_date****

(customer -> before / insert)

```
SET NEW.create_date = NOW()
```

customer에서 insert 실행 전
create_date에 now()값을 넣어준다.

****payment_date* ***

(payment -> before / insert)

```
SET NEW.payment_date = NOW()
```

payment에서 insert 실행 전
payment_date에 now()값을 넣어준다.

****rental_date****

(rental -> before / insert)

```
SET NEW.rental_date = NOW()
```

rental에서 insert 실행 전
rental_date에 now()값을 넣어준다.

****del_film****

(film -> after / delete)

```
BEGIN
    DELETE FROM film_text WHERE film_id = old.film_id;
END
```

film에서 delete 실행 후
삭제 전 film_id와 같은 film_id를 가진 행을 film_text에서 지운다.

****ins_film****

(film -> after / insert)

```
BEGIN
    INSERT INTO film_text (film_id, title, description)
        VALUES (new.film_id, new.title, new.description);
END
```

film에서 insert 실행 후
film_text의 film_id, title, description 값을 입력한다.
-> film 컬럼 중 film_id, title, description이 있음. 입력된 값을 film_text에 그대로 가져온다.

****upd_film****

(film -> after / update)

```
        BEGIN
            IF (old.title != new.title) OR (old.description != new.description) OR
(old.film_id != new.film_id)
            THEN
                UPDATE film_text
                    SET title=new.title,
                        description=new.description,
                        film_id=new.film_id
                WHERE film_id=old.film_id;
            END IF;
          END
```

film에서 update 실행 후
title, description, film_id 중 하나라도 수정될 경우 셋의 값을 film의 값과 동일하게 수정한다.

*********트리거 기본 내용 설명 후 시간 부족할 때 추가로 해볼만 한 것***********

(*. 트리거 활용)
임시저장, 데이터 복구, 채팅 로그 등 다양한 곳에 활용 가능

(*. 트리거의 오류)
트리거에 오류가 있거나, 엔진이 트랜잭션을 지원하거나, 트리거가 여러행에 영향을 미치는 경우 트리거는 실행하지 않으며 트리거를 발생시킨 원문 sql도 실행되지 않는다.
트리거 오류 o -> 트랜잭션 지원o or before 트리거 -> 트리거 실행x, 원문sql 실행x
트랜잭션 지원x, after 트리거 -> 원문sql 실행o, 트리거 실행o

(*. 트리거의 장단점)