3. 트리거와 주장

트리거 : 데이터베이스 운영에서 데이터 **무결성**을 유지하기 기능!
트리거는 데이터베이스에 어떠한 일이 일어나면 **자동으로 실행**되는 개체이다.

SQL 서버나 데이터베이스에 어떤 **이벤트**가 발생했을 때, 정의된 **트리거가 수행**될 수 있다.
트리거는 제약조건과 더불어 데이터 **무결성**을 위해서 사용할 수 있는 기능이다.
트리거는 잘못된 갱신을 예방할 수 있는 특수 제약조건으로 사용될 수 있다.
트리거는 사건(삽입, 삭제, 갱신) 발생 **후**가 아닌 사건 발생 **전**에도 활성화 될 수 있다.

// 트리거 종류

DDL 트리거	• SQL 서버나 데이터베이스에서 DDL 이벤트에 반응하여 작동한다. • DDL 트리거는 Create, Alter, Drop 문을 실행할 때 작동된다. • DDL 이벤트 : Create_Table, Alter_Table, Drop_Table 등 • DDL 트리거는 데이터베이스에 대한 감시, 통제 작업 등에 활용된다.
DML 트리거	• 테이블이나 뷰와 관련되어 DML 이벤트에 반응하여 작동한다. • 트리거 발생 DML : insert, update, delete • 테이블에 어떤 데이터가 **삽입/삭제/갱신**될 때, DML 트리거가 발생될 수 있다. • Select는 이벤트가 발생하지 않는다.**(Select는 트리거 발생 안됨)**
LOGON 트리거	• 사용자가 SQL 서버에 로그온(로그인)할 때 작동하는 트리거이다.

〈DDL 트리거 생성 형식〉

```
Create Trigger 트리거명 On {Database | All Server} [With Encryption]
{For 이벤트 타입 | 이벤트 그룹}
As 실행할 Sql 문장들
```

↓예

```
Create Trigger safety On Database
For Drop_Table, Alter_Table
As Print 'You must disable trigger "safety" to drop or alter tables!'
```

예제 1	사원의 급여를 최대 10%까지만 인상하기 위한 트리거

사원

사번	이름	소속	상사	급여
1	홍길동	10	NULL	500
2	이몽룡	20	100	400
3	성춘향	20	200	300
4	박대감	30	100	350

```
Create Trigger 급여_인상                    //트리거 생성(트리거명 : 급여_인상)
After Update Of 급여 On 사원                //사원 테이블에서 급여 필드를 갱신하면 실행
Referencing Old Row As O, New Row As N //새로운 값은 N, 이전 값은 O
For Each Row
When (N.급여 > O.급여 × 1.1)                //새로운 급여가 기존 급여보다 110% 초과하면
      Update 사원 Set 급여 = O.급여 × 1.1   //급여를 강제로 110% 인상 갱신하라.
      Where 사원.사번 = O.사번;
```

- 사원 급여를 최대 10%까지만 인상하는데 잘못 처리해서 인상률이 10% 초과하게 되면
- 급여_인상 트리거가 자동으로 실행되어 급여를 최대 10%까지만 인상시킨다.

New	• 트리거가 생성하는 **임시테이블**, 변경할 **새로운 값**을 임시로 저장, **변경할 테이블에 적용** • New 테이블은 Insert, Update 작업에서 변경할 **새로운 값**을 잠깐 저장한다. • Insert(새로운 값) → New 테이블(새로운 값 저장) → 변경할 테이블(새로운 값 입력)
Old	• 트리거가 생성하는 **임시테이블**, 변경되기 **이전 값**을 임시로 저장 • Old 테이블은 Delete, Update 작업에서 **삭제, 변경**되기 **이전 값**을 임시로 저장한다. • Delete(이전 값) → Old 테이블(이전 값 저장) → 변경할 테이블(이전 값 삭제)

// Update는 Old와 New 모두 생성된다.

Update(이전 값, 새로운 값)

↓

Old 테이블(이전 값 저장)
New 테이블(새로운 값 저장)

↓

변경할 테이블(이전 값 삭제, 새로운 값 입력)

// New와 Old 사용 가능 여부

이벤트	New	Old
Insert	○	×
Delete	×	○
Update	○	○

- 트리거 작동에서 새로 입력, 변경되는 새로운 값을 참조하기 위해서는 **New 테이블을 참조하고**
- 트리거 작동에서 변경되기 이전의 값을 참조하기 위해서는 **Old 테이블을 참조**한다.

예제 2	영업실적 테이블에 새로운 레코드 삽입시마다 영업건수 테이블의 건수 값이 1 증가

----<보기>----------

영업실적

사번	이름	수주량

영업건수

건수
0

초기 상태

↓레코드 추가

영업실적

사번	이름	수주량
1	홍재연	100

영업건수

건수
1

영업건수 1 증가

↓레코드 추가

영업실적

사번	이름	수주량
1	홍재연	100
2	이순신	200

영업건수

건수
2

영업건수 1 증가

↓레코드 추가

영업실적

사번	이름	수주량
1	홍재연	100
2	이순신	200
3	홍하은	300

영업건수

건수
3

영업건수 1 증가

```
Create Trigger 영업_건수                  //영업_건수 트리거 생성
After Insert On 영업실적                  //이벤트 : 영업실적 테이블에 새로운 레코드 삽입
For Each Row                              //영업실적 테이블에 새로운 레코드 삽입시 트리거 수행
Update 영업건수 Set 건수 = 건수 + 1;      //영업건수 테이블의 건수를 강제로 1 증가
```

After	변경 사항이 데이터베이스에 적용된 **후**에 트리거를 수행한다.
Before	변경 사항이 데이터베이스에 적용되기 **전**에 미리 트리거를 수행한다.

예제 3	영업실적 테이블에 레코드 삽입시마다 누적통계 테이블의 누적건수와 누적수주량 갱신

----<보기>----------

영업실적

사번	이름	수주량

누적통계

누적건수	누적수주량
0	0

초기 상태

↓레코드 추가

영업실적

사번	이름	수주량
1	홍재연	100

누적통계

누적건수	누적수주량
1	100

누적건수 및 누적수주량

↓레코드 추가

영업실적

사번	이름	수주량
1	홍재연	100
2	이순신	200

누적통계

누적건수	누적수주량
2	300

누적건수 및 누적수주량

↓레코드 추가

영업실적

사번	이름	수주량
1	홍재연	100
2	이순신	200
3	홍하은	300

누적통계

누적건수	누적수주량
3	600

누적건수 및 누적수주량

```
Delimiter $$                          //트리거 시작, 구분기호
Create Trigger 영업_누적통계           //트리거 생성 : 영업_누적통계
Before Insert On 영업실적              //영업실적 테이블에 새로운 레코드 삽입 전에
For Each Row
Begin                                 //여러 개의 구문을 묶기 위한 것
  Update 누적통계 Set 누적건수 = 누적건수 + 1;                  //영업건수 누적
  Update 누적통계 Set 누적수주량 = 누적수주량 + New.수주량;     //영업수주량 누적
End $$
Delimiter;                            //트리거 종료
```

• New 테이블은 Insert 작업에서 삽입될 **새로운 값**을 임시 저장한다.

참고	MariaDB 10.2.3까지는 하나의 테이블은 하나의 이벤트/타이밍 조합에 대해 **오직 한 가지 트리거**만을 **정의**할 수 있다.

예제 4	할인 테이블의 특정 품명의 할인율 필드가 갱신되면 상품 테이블의 해당 품명의 할인율 필드도 똑같이 갱신되도록 한다.

----<보기>----------

할인

품명	할인율
마우스	0.10
키보드	0.20
소리통	0.30
마이크	0.40

상품

품명	가격	할인율
마우스	30,000	0.10
키보드	20,000	0.20
소리통	30,000	0.30
마이크	40,000	0.40

↓할인 테이블의 마이크 할인율을 0.45로 수정

할인

품명	할인율
마우스	0.10
키보드	0.20
소리통	0.30
마이크	0.45

상품

품명	가격	할인율
마우스	30,000	0.10
키보드	20,000	0.20
소리통	30,000	0.30
마이크	40,000	0.45

상품 테이블의 마이크 할인율 0.45로 자동 수정

```
Delimiter $$                          //트리거 시작, 구분기호
Create Trigger 할인율_일치             //트리거 생성
After Update On 할인                  //이벤트 : 할인 테이블 갱신 후에
For Each Row
Begin
  If New.할인율 != Old.할인율 Then     //할인율이 새로 갱신되면
    Update 상품 Set 할인율 = New.할인율 Where New.품명 = 상품.품명;
  End If;                                          ↳ 할인율이 갱신된 품명
End $$          // 상품 테이블의 해당 품명의 할인율을 새로 갱신된 할인율로 수정
Delimiter;
```

// 분석

```
Update 상품 Set 할인율 = New.할인율 Where New.품명 = 상품.품명;
                                                  ↳ 상품 테이블의 품명(당연)
                                  ↓
                  New.품명은 할인 테이블의 품명
```

예제 5	주문서 테이블에서 특정 주문 내용이 삭제되면(취소되면) 삭제된 상품 주문 내용이 주문취소 테이블에 자동으로 삽입(백업)되도록 한다.

----<보기>----------

주문서

품명	가격	수량
마우스	30,000	10
키보드	20,000	20
소리통	30,000	30

주문취소

품명	가격	수량

– 빈상태

↓주문서 테이블의 키보드 삭제(취소)

주문서

품명	가격	수량
마우스	30,000	10
소리통	30,000	30

주문취소

품명	가격	수량
키보드	20,000	20

↓주문서 테이블의 마우스 삭제(취소)

주문서

품명	가격	수량
소리통	30,000	30

주문취소

품명	가격	수량
키보드	20,000	20
마우스	30,000	10

```
Delimiter $$                          //트리거 시작, 구분기호
Create Trigger 주문취소_자동백업       //트리거 생성
Before Delete On 주문서               //이벤트 : 주문서 테이블 내용 삭제 전에
For Each Row
Begin
  Declare 품명_백업 char;             //변수 선언
  Declare 가격_백업 Int;
  Declare 수량_백업 Int;
  Set 품명_백업 = Old.품명;           //백업할 내용 대입
  Set 가격_백업 = Old.가격;
  Set 수량_백업 = Old.수량;
  Insert Into 주문취소 Values(품명_백업, 가격_백업, 수량_백업);
End $$                    ↓
Delimiter;        주문서 테이블에서 삭제된 내용을 주문취소 테이블에 삽입
```

참고	위의 내용은 MySQL과 mariaDB에서 실제로 실습한 결과이다.

// 트리거(trigger)와 주장(assertion)

트리거 (trigger)	• 트리거는 명시된 조건을 만족하면 특정한 동작을 **자동으로 수행**할 수 있도록 한다. • 트리거는 데이터베이스 **무결성**을 유지하기 위한 일반적이고 강력한 도구이다. • 트리거는 **제약조건을 위반했을 때 수행할 동작**을 명시하는 것이다. • 트리거는 **이벤트-조건-동작** 규칙이라고도 한다.
주장 (assertion)	• 주장은 **제약조건을 위반하는 연산은 수행되지 않도록** 하는 것이다. • 주장은 트리거보다 좀 더 **일반적인 무결성 제약조건**에 적용한다. • 주장은 2개 이상의 테이블에 영향을 미치는 제약조건 명시에 사용될 수 있다.

// 트리거 사용 및 특징

- 트리거는 직접 실행시킬 수는 없다.
- 트리거는 이벤트가 발생한 경우에만 내부적으로 자동 실행된다.
- 트리거는 시스템 성능을 저하시키는 요인이 될 수 있다.
- 트리거는 반드시 필요한 곳에서만 사용하여야 한다.

예제 1	사원 테이블에서 급여 〉 5000이 되지 않도록 주장하는 SQL문

사원

사번	이름	급여
1	순돌	1000
2	순이	2000
3	철수	3000
4	순돌	4000

↓주장에 대한 SQL문

```
Create Assertion 최대_급여          //주장 제약조건 이름
As Check                            //Check 다음에 제약조건 명시
(Not Exists                         //제약조건 명시
    (Select * From 사원 Where 급여 > 5000)
);
```

- 사원 테이블에서 급여 〉 5000이 되지 않도록 주장하는 SQL문
- 주장은 대상 테이블과 관련된 SQL의 갱신문 수행될 때, 주장에 대한 SQL문이 검사된다.
- 주장의 **제약조건**을 검사하여 **참**이면 테이블 **수정**을 **허용**한다.
- 예제 1에서 급여 〉 5000이 되면 **제약조건**은 **거짓**이므로 **테이블 수정**을 **허용하지 않는다.**

예제 2	수강 테이블에 새로운 레코드 추가하는 경우

학생

학번	이름	학년
1	순돌	2
2	순이	1
3	철수	2

수강

학번	*과목*	점수
1	디비	90
1	소공	80
2	디비	90

주장(assertion)	학생 테이블에 없는 학생의 학번은 수강 테이블에 나타나지 않도록 한다. 즉, 학생 테이블에 없는 튜플은 수강 테이블에 존재하지 않아야 한다.

↓주장에 대한 SQL문

```
Create Assertion 학생_무결성                    //주장 제약조건 이름
As Check                                        //Check 다음에 제약조건 명시
(Not Exists                                     //제약조건 명시
    (Select * From 수강 Where 학번 Not In (Select 학번 From 학생))
);
```

- 주장은 대상 테이블과 관련된 SQL의 갱신문 수행될 때, 주장에 대한 SQL문이 검사된다.
- 주장의 제약조건을 검사하여 **참**이면 테이블 **수정**을 허용한다.
- 즉, 주장에 대한 SQL문의 제약조건에 따라 연산이 수행되지 않을 수 있다.
- 주장은 불필요할 경우 DROP 문으로 제거할 수 있다.

구문	Create Assertion 이름 As Check 제약조건;

기출문제 분석

1. 다음에서 설명하는 데이터베이스 용어는? [2023년 국가 7급]

> ○ 데이터베이스의 갱신이 발생할 때 DBMS가 자동으로 수행하는 사용자 정의 프로시저이다.
> ○ 무한 호출에 따라 실행의 종료를 보장할 수 없는 문제가 있다.
> ○ 무결성 제약조건을 유지하기 위하여 데이터베이스 갱신을 모니터링하고, 데이터베이스 갱신을 전파한다.

① assertion ② catalog
③ check ④ trigger

♤ trigger

- 트리거는 데이터베이스에 어떠한 일이 일어나면 **자동으로 실행**되는 개체이다.
- 트리거는 데이터베이스 갱신이 발생할 때 DBMS가 자동으로 수행하는 사용자 프로시저이다.
- SQL 서버나 데이터베이스에 어떤 **이벤트**가 발생했을 때, 정의된 **트리거가 수행**될 수 있다.
- 트리거는 제약조건과 더불어 데이터 **무결성**을 위해서 사용할 수 있는 기능이다.
- 트리거는 잘못된 갱신을 예방할 수 있는 특수 제약조건으로 사용될 수 있다.
- 트리거는 사건(삽입, 삭제, 갱신) 발생 **후**가 아닌 사건 발생 **전**에도 활성화 될 수 있다.

사원

사번	이름	소속	상사	급여
1	홍길동	10	NULL	500
2	이몽룡	20	100	400
3	성춘향	20	200	300
4	박대감	30	100	350

```
Create Trigger 급여_인상                    //트리거 생성(트리거명 : 급여_인상)
After Update Of 급여 On 사원                //사원 테이블에서 급여 필드를 갱신하면 실행
Referencing Old Row As O, New Row As N //새로운 값은 N, 이전 값은 O
For Each Row
When (N.급여 > O.급여 × 1.1)                //새로운 급여가 기존 급여보다 110% 초과하면
      Update 사원 Set 급여 = O.급여 × 1.1   //급여를 강제로 110% 인상 갱신하라.
      Where 사원.사번 = O.사번;
```

- 사원 급여를 최대 10%까지만 인상하는데 잘못 처리해서 인상률이 10% 초과하게 되면
- 급여_인상 트리거가 자동으로 실행되어 급여를 최대 10%까지만 인상시킨다.

정답 : ④

2. SQL 트리거(trigger)에 대한 설명으로 옳지 않은 것은? [2009년 국가 7급]

① SQL 트리거는 사건(event), 조건(condition), 동작(action) 부분으로 구성된다.
② 동작은 트리거시키는 사건의 전(before)이나 후(after)에 실행될 수 있다.
③ 동작은 규칙이 트리거되고 트리거 사건 발생 시 조건이 만족될 때 실행된다.
④ 트리거시키는 가능한 사건으로는 SELECT, UPDATE, INSERT, DELETE 등이 있다.

♤ SQL 트리거(trigger)

• 트리거시키는 가능한 사건으로는 SELECT, UPDATE, INSERT, DELETE 등이 있다.(×)
→ SELECT는 트리거를 발생시키지 않는다.(데이터베이스 내용이 변경되지 않으므로)
→ UPDATE, INSERT, DELETE의 이벤트가 발생할 때 트리거 작동된다.

• SQL 서버나 데이터베이스에 어떤 이벤트가 발생했을 때, 트리거가 발생될 수 있다.
• 트리거는 데이터베이스에 어떠한 일이 일어나면 자동으로 실행되는 개체이다.
• 트리거는 제약조건과 더불어 데이터 무결성을 위해서 사용할 수 있는 기능이다.
• 트리거는 사건(삽입, 삭제, 갱신) 발생 후가 아닌 사건 발생 전에도 활성화 될 수 있다.
→ 이런 트리거는 잘못된 갱신을 예방할 수 있는 특수 제약조건으로 사용될 수 있다.

// 사원의 급여를 최대 10%까지만 인상하기 위한 트리거

사원

사번	이름	소속	상사	급여
1	홍길동	10	NULL	500
2	이몽룡	20	100	400
3	성춘향	20	200	300
4	박대감	30	100	350

부서

부서번호	부서명	부서장
10	사장실	100
20	총무부	200
30	영업부	400

```
Create Trigger 직원                          //트리거 생성(트리거 이름 : 직원)
After Update Of 급여 On 사원                 //사원 테이블에서 급여 필드를 갱신하면 실행
Referencing Old Row As O, New Row As N       //새로운 값은 N, 이전 값은 O
For Each Row
When (N.급여 > O.급여 × 1.1)                 //새로운 급여가 기존 급여보다 110% 초과하면
      Update 사원 Set 급여 = O.급여 × 1.1    //급여를 강제로 110% 인상 갱신하라.
      Where 사원.사번 = O.사번;
```

정답 : ④

3. 다음 관계형 데이터베이스의 세 가지 기능적 요소에 대한 설명에서 ㉠~㉢에 들어갈 용어를 바르게 연결한 것은? [2019년 국가 7급]

○ (㉠)는(은) SQL에서 삽입, 삭제, 갱신과 같은 데이터 변경문을 실행할 때 미리 명시된 조건을 만족하는 경우 특정한 동작을 자동으로 수행할 수 있도록 한다.
○ (㉡)는(은) 데이터베이스 내에 존재하는 작업순서가 정해진 수행 단위로서 DBMS에서 컴파일된 후 실행된다.
○ (㉢)는(은) 데이터베이스에서 데이터를 신속하게 탐색할 수 있도록 만든 데이터 구조이다.

	㉠	㉡	㉢
①	인덱스	트리거(trigger)	주장(assertion)
②	주장	인덱스	저장 프로시저(stored procedure)
③	주장	인덱스	트리거
④	트리거	저장 프로시저	인덱스

♤ **관계형 데이터베이스 기능**

// 트리거(trigger)
- SQL 서버나 데이터베이스에 어떤 이벤트가 발생했을 때, 트리거가 발생될 수 있다.
- 트리거는 명시된 조건을 만족하는 경우 특정한 동작을 자동으로 수행할 수 있도록 한다.
- 트리거는 데이터베이스 무결성을 유지하기 위한 일반적이고 강력한 도구이다.
- 트리거는 **제약조건을 위반했을 때 수행할 동작**을 명시하는 것이다.

// 주장(assertion)
- 주장은 **제약조건을 위반하는 연산은 수행되지 않도록** 하는 것이다.
- 주장은 트리거보다 좀 더 일반적인 무결성 제약조건에 적용한다.
- 주장은 2개 이상의 테이블에 영향을 미치는 제약조건을 명시하기 위해 사용될 수 있다.

// 저장 프로시저
- 저장 프로시저는 데이터베이스 내에 존재하는 작업순서가 정해진 수행 단위이다.
- 저장 프로시저는 일련의 쿼리를 하나의 함수처럼 실행하기 위한 쿼리 집합이다.
- 저장 프로시저는 DBMS에서 컴파일된 후 데이터베이스시스템 내부에 저장되어 실행된다.
- 저장 프로시저는 데이터베이스 서버와 같은 프로세스 공간에서 실행된다.

// 인덱스
- 인덱스는 데이터베이스에서 데이터를 신속하게 탐색할 수 있도록 만든 데이터 구조이다.
- 인덱스는 하나의 필드 또는 여러 필드를 기반으로 하여 인덱스를 만들 수 있다.

정답 : ④

4. 무결성 규정(integrity rule)에 대한 설명으로 가장 옳지 않은 것은? [2018년 서울 7급]

① 트리거(trigger)를 사용하면 하나의 릴레이션이 변경될 때, 다른 릴레이션도 변경될 수 있다.
② 무결성 규정은 시스템 카탈로그(system catalog)나 데이터 사전에 저장된다.
③ 갱신 연산을 할 때, 릴레이션 무결성 규정은 트랜잭션이 완전히 수행된 이후에 적용된다.
④ SQL에서 CREATE 문으로 도메인을 생성할 때, CHECK 구문을 통해 도메인 제약조건을 지정할 수 있다.

♤ **무결성 규정(integrity rule)**

- 갱신 연산을 할 때, 릴레이션 무결성 규정은 트랜잭션이 완전히 수행된 이후에 적용된다.**(×)**
  → 무결성 규정은 트랜잭션 수행 도중에 제약조건에 위배되면 적용된다.
  → 예 : 제약조건에 위배되면, 데이터가 삽입되지 않는다.

정답 : ③

5. 데이터베이스 트리거(trigger)에 대한 설명으로 옳지 않은 것은? [2021년 국가 7급]

① 테이블에서 이벤트 발생 시 자동으로 반응해 실행되는 작업이다.
② 트리거는 데이터베이스의 무결성을 유지하기 위한 도구이다.
③ 행-수준 트리거(row-level trigger)는 For Each Statement절을 사용하여 표시한다.
④ 이벤트가 발생한 이후 실행되는 After 트리거와 이벤트가 발생하기 전에 실행되는 Before 트리거가 있다.

♤ **트리거**

- **행-수준** 트리거(row-level trigger)는 **For Each Statement**절을 사용하여 표시한다.**(×)**
  → **행-수준** 트리거 : **For Each Row**
- 트리거는 제약조건과 더불어 데이터 **무결성**을 위해서 사용할 수 있는 기능이다.
- 트리거는 잘못된 갱신을 예방할 수 있는 특수 제약조건으로 사용될 수 있다.
- 트리거는 **사건**(삽입, 삭제, 갱신) **발생 후**가 아닌 사건 **발생 전**에도 활성화 될 수 있다.

정답 : ③

6. 다음은 데이터베이스의 무결성을 유지하기 위한 강력한 도구의 하나인 트리거(trigger)에 대한 설명이다. 가장 적절하지 않은 것은? [2023년 군무 7급]

① 트리거는 각 테이블마다 한 개의 트리거만 가질 수 있으므로 트리거들이 연쇄적으로 활성화되는 복잡한 경우를 미리 방지할 수 있다.
② 트리거는 제약조건과 유사하지만 어떤 이벤트가 발생했을 때 조건이 참이 되면 트리거와 연관된 동작이 수행되고 그렇지 않으면 아무 동작도 수행되지 않는다.
③ 트리거의 작동은 저장 프로시저와 비슷하지만 직접 실행시킬 수는 없고, 저장 프로시저와는 달리 입출력 매개변수를 사용할 수 없다.
④ 트리거를 사용하면 누군가 테이블의 행을 고의 또는 실수로 삭제했을 경우에 행이 삭제되는 순간에 삭제된 내용, 시간, 사용자 등을 기록해 놓을 수 있다.

♤ **트리거**

• 트리거는 **각 테이블**마다 **한 개**의 **트리거만** 가질 수 있으므로 트리거들이 연쇄적으로 활성화되는 복잡한 경우를 미리 방지할 수 있다.(×)
→ 테이블은 **여러 개**의 **트리거**를 가질 수 있다
→ 그런데, 하나의 테이블에 대해 하나의 **트리거**만을 정의할 수 있는 DBMS도 있다

참고	MariaDB 10.2.3까지는 하나의 테이블은 하나의 이벤트/타이밍 조합에 대해 **오직 한 가지 트리거**만을 **정의**할 수 있다.(**트리거**를 많이 적용하면 **복잡성** 초래)

예제	사원의 급여를 최대 10%까지만 인상하기 위한 트리거

사원

사번	이름	소속	상사	급여
1	홍길동	10	NULL	500
2	이몽룡	20	100	400
3	성춘향	20	200	300
4	박대감	30	100	350

```
Create Trigger 급여_인상                    //트리거 생성(트리거명 : 급여_인상)
After Update Of 급여 On 사원                //사원 테이블에서 급여 필드를 갱신하면 실행
Referencing Old Row As O, New Row As N //새로운 값은 N, 이전 값은 O
For Each Row
When (N.급여 > O.급여 × 1.1)                //새로운 급여가 기존 급여보다 110% 초과하면
      Update 사원 Set 급여 = O.급여 × 1.1   //급여를 강제로 110% 인상 갱신하라.
      Where 사원.사번 = O.사번;
```

정답 : ①

7. 다음 SQL 트리거 명령문은 직원(EMPLOYEE)의 급여(Salary)를 갱신하면 그 직원이 근무하는 부서(DEPARTMENT)의 전체 직원급여(Total_sal)도 자동으로 갱신하는 기능을 수행한다. 빈칸에 들어갈 내용으로 적절한 것은? (단, Dno는 부서번호이다) [2014년 국가 7급]

```
-----------------------------------------------------
CREATE TRIGGER SALARY_TRIGGER
[                                  ]
WHEN (N.Dno IS NOT NULL)
UPDATE DEPARTMENT
SET Total_sal = Total_sal + N.Salary - O.Salary
WHERE Dno = N.Dno;
-----------------------------------------------------
```

① AFTER UPDATE OF Salary ON EMPLOYEE
REFERENCING OLD ROW AS O, NEW ROW AS N FOR EACH ROW

② BEFORE UPDATE OF EMPLOYEE
REFERENCING OLD ROW AS O, NEW ROW AS N FOR EACH ROW

③ AFTER UPDATE OF EMPLOYEE
REFERENCING OLD TABLE AS O, NEW TABLE AS N FOR EACH STATEMENT

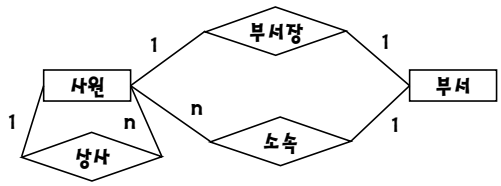
④ BEFORE UPDATE OF Salary ON EMPLOYEE
REFERENCING OLD TABLE AS O, NEW TABLE AS N FOR EACH STATEMENT

♤ **트리거 형식**

```
CREATE TRIGGER SALARY_TRIGGER              //트리거 생성(트리거명 : SALARY_TRIGGER)
[
  AFTER UPDATE OF Salary ON EMPLOYEE    //이벤트 : 직원의 급여(Salary)를 갱신한 후에
  REFERENCING OLD ROW AS O, NEW ROW AS N
  FOR EACH ROW
]
WHEN (N.Dno IS NOT NULL)                    //조건 : 부서번호(N.Dno0가 NULL이 아니면
UPDATE DEPARTMENT
SET Total_sal = Total_sal + N.Salary - O.Salary   //부서(DEPARTMENT)의 전체 급여 갱신
WHERE Dno = N.Dno;
```

정답 : ①

8. 다음과 같이 사원과 부서 테이블을 이용하는 회사가 직원의 급여를 인상할 때 오류를 방지하기 위해 〈설명〉과 같은 SQL99 표준에 따른 트리거를 정의하여 사용 중이다. 이와 같은 조건에서 직원들의 급여 인상을 위한 〈보기〉의 질의를 수행하였다. 급여 인상의 결과를 보기 위해 질의 "SELECT 사번, 이름, 급여 FROM 사원"을 수행한 결과로 옳은 것은? [2016년 국가 7급]



사원

사번	이름	소속	상사	급여
100	홍길동	10	NULL	500
200	이몽룡	20	100	400
300	성춘향	20	200	300
400	박대감	30	100	350

부서

부서번호	부서명	부서장
10	사장실	100
20	총무부	200
30	영업부	400

---------<설 명>----------

```
CREATE TRIGGER Employment
AFTER UPDATE OF 급여 ON 사원
REFERENCING OLD ROW AS O, NEW ROW AS N
FOR EACH ROW
WHEN (N.급여 > O.급여 * 1.1)
      UPDATE 사원 SET 급여 = O.급여 * 1.1
      WHERE 사원.사번 = O.사번;
```

---------<보 기>----------

```
UPDATE 사원 SET 급여 = 급여 * 1.2
          WHERE 사번 IN (SELECT 부서장 FROM 부서);
UPDATE 사원 SET 급여 = 급여 * 1.05
          WHERE 사번 != ALL (SELECT 부서장 FROM 부서);
UPDATE 사원 SET 급여 = 급여 * 1.1 WHERE 사번 = 100;
```

---

① 사원

사번	이름	급여
100	홍길동	550
200	이몽룡	420
300	성춘향	315
400	박대감	380

② 사원

사번	이름	급여
100	홍길동	550
200	이몽룡	420
300	성춘향	330
400	박대감	380

③

사원

사번	이름	급여
100	홍길동	605
200	이몽룡	440
300	성춘향	315
400	박대감	385

④

사원

사번	이름	급여
100	홍길동	605
200	이몽룡	440
300	성춘향	330
400	박대감	385

♤ 급여 인상 트리거

// CREATE 문장 분석(급여를 최대 10%까지만 인상하기 위한 것)

```
Create Trigger employment
After Update Of 급여 On 사원                    //사원 테이블에서 급여 필드를 갱신하면 실행
Referencing Old Row As O, New Row As N           //새로운 값은 N, 이전 값은 O
For Each Row
When (N.급여 > O.급여 × 1.1)                     //새로운 급여가 기존 급여보다 110% 초과하면
      Update 사원 Set 급여 = O.급여 × 1.1        //급여를 강제로 110% 인상 갱신하라.
      Where 사원.사번 = O.사번;
```

↓갱신 작업 수행

```
■ Update 사원 Set 급여 = 급여 × 1.2              //부서장 급여는 20%인상
  Where 사번 In (Select 부서장 From 부서);
■ Update 사원 Set 급여 = 급여 × 1.05             //부서장 아닌 사람 급여는 5%인상
  Where 사번 != All (Select 부서장 From 부서);
■ Update 사원 Set 급여 = 급여 × 1.1              //사번이 100인 사람 급여는 추가로 10%인상
  Where 사번 = 100;
```

// 실행 결과 분석

- UPDATE 실행은 위와 같지만,
- 〈설명〉의 트리거 수행으로 인하여 **부서장 급여**는 20%가 아닌 10%만 인상된다.
  → 부서장인 홍길동, 이몽룡, 박대감은 급여가 10% 인상된다.
  → 홍길동(550), 이몽룡(440), 박대감(385)
- **부서장이 아닌 사람 급여**는 5%인상된다. 성춘향 5% 인상(315)
- 그리고, **사번 = 100**인 홍길동은 추가로 급여가 10% 인상된다. 최종적으로 홍길동(605)

정답 : ③

9. 할인율 누계 평균이 30%를 초과했을 경우, 신규 주문을 입력할 때 주문 제품에 대한 할인율을 10% 하향 설정하려고 한다. 아래 주문 테이블의 정의를 참조하여, 그 아래 SQL 트리거 문장의 빈칸 (ㄱ), (ㄴ)에 각각 들어갈 내용으로 가장 적절한 내용은? [2021년 군무원 7급]

```
-------------------------------------------------------------
CREATE TABLE 주문(
  OrderID INT PRIMARY KEY,
  날짜 DATE,
  제품ID INT,
  가격 INT,
  할인율 NUMERIC(7, 5) DEFAULT 0.0
);
-------------------------------------------------------------
CREATE TRIGGER Check_Discount_Rate (   ㄱ   ) ON 주문
FOR EACH ROW
BEGIN
  SET @value=(SELECT AVG(할인율) FROM 주문);
  IF @value>0.3 THEN
    SET (   ㄴ   )
  END IF;
END
-------------------------------------------------------------
```

① (ㄱ) BEFORE INSERT (ㄴ) NEW.할인율 = OLD.할인율 * 0.9;
② (ㄱ) BEFORE INSERT (ㄴ) NEW.할인율 = NEW.할인율 * 0.9;
③ (ㄱ) AFTER INSERT (ㄴ) NEW.할인율 = OLD.할인율 * 0.9;
④ (ㄱ) AFTER INSERT (ㄴ) NEW.할인율 = NEW.할인율 * 0.9;

♤ SQL – 트리거

```
CREATE TRIGGER Check_Discount_Rate (ㄱ.BEFORE INSERT) ON 주문
FOR EACH ROW                          ↳ 신규 주문을 입력할 때(입력 전에)
BEGIN
  SET @value=(SELECT AVG(할인율) FROM 주문);
  IF @value>0.3 THEN                  //신규 주문시 할인율 누계 평균이 30% 초과할 때
    SET (ㄴ.NEW.할인율 = NEW.할인율 * 0.9;)
  END IF;          ↳ 신규 주문시 할인율 누계 평균이 30% 초과할 때, 할인율을 10% 하향 설정
END
```

정답 : ②

10. 다음 SQL 구문에 대한 설명으로 옳지 않은 것은? [2020년 국가 7급]

```
CREATE ASSERTION NO_DESIGNERS_IN_SEOUL
AS CHECK
(NOT EXISTS
   (SELECT * FROM EMPLOYEE E, DEPARTMENT D
   WHERE E.DEPTNO = D.DEPTNO AND E.JOB = 'DESIGNER' AND D.LOC = 'SEOUL')
);
```

① 명시된 이벤트가 발생할 때마다 DBMS가 자동적으로 수행하는 구문이다.
② 대상 테이블과 관련된 SQL의 갱신문 수행 시 위의 구문이 검사된다.
③ 위의 SQL이 불필요할 경우 DROP문으로 제거할 수 있다.
④ 위의 SQL에서 명세하고 있는 조건에 따라 DB연산이 수행되지 않을 수 있다.

♤ SQL 구문 – ASSERTION

• CREATE ASSERTION : 주장 생성(일반적인 제약조건을 명시)

```
CREATE ASSERTION NO_DESIGNERS_IN_SEOUL    //제약조건 이름
AS CHECK                                  //Check 다음에 제약조건 명시
( NOT EXISTS                              //제약조건 명시
    ( SELECT * FROM EMPLOYEE E, DEPARTMENT D
      WHERE E.DEPTNO = D.DEPTNO
        AND E.JOB = 'DESIGNER' AND D.LOC = 'SEOUL' )
);
```

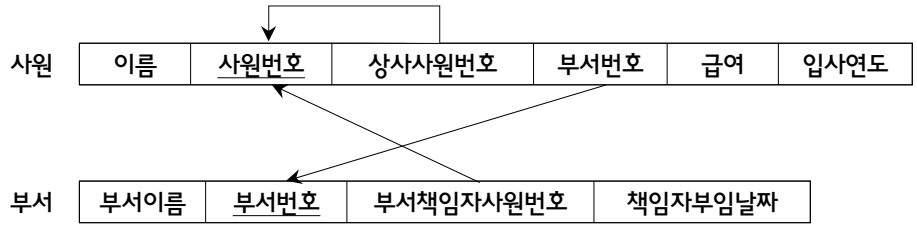
// 트리거(trigger)와 주장(assertion)

트리거 (trigger)	• 트리거는 명시된 조건을 만족하면 특정한 동작을 **자동으로 수행**할 수 있도록 한다. • 트리거는 데이터베이스 **무결성**을 유지하기 위한 일반적이고 강력한 도구이다. • 트리거는 **제약조건을 위반했을 때 수행할 동작**을 명시하는 것이다.
주장 (assertion)	• 주장은 **제약조건을 위반하는 연산은 수행되지 않도록** 하는 것이다. • 주장은 트리거보다 좀 더 **일반적인 무결성 제약조건**에 적용한다. • 주장은 2개 이상의 테이블에 영향을 미치는 제약조건 명시에 사용될 수 있다.

• 주장은 대상 테이블과 관련된 SQL의 갱신문 수행될 때, 주장에 대한 SQL문이 검사된다.
• 주장의 **제약조건**을 검사하여 **참**이면 테이블 **수정**을 **허용**한다.
• 즉, 주장에 대한 SQL문의 제약조건에 따라 연산이 수행되지 않을 수 있다.

정답 : ①

11. 다음 데이터베이스 스키마에 대한 설명으로 옳지 않은 것은? (단, 밑줄이 있는 속성은 그 릴레이션의 기본키를, 화살표는 외래키 관계를 의미한다) [2015년 지방 9급]



① 외래키는 동일한 릴레이션을 참조할 수 있다.
② 사원 릴레이션의 부서번호는 부서 릴레이션의 부서번호 값 중 하나 혹은 널이어야 한다는 제약조건은 참조무결성을 의미한다.
③ 신입사원을 사원 릴레이션에 추가할 때 그 사원의 사원번호는 반드시 기존 사원의 사원번호와 같지 않아야 한다는 제약조건은 제1정규형의 원자성과 관계있다.
④ 부서 릴레이션의 책임자부임날짜는 반드시 그 부서책임자의 입사연도 이후이어야 한다는 제약조건을 위해 트리거(trigger)와 주장(assertion)을 사용할 수 있다.

♤ **데이터베이스 스키마**

- 신입사원을 사원 릴레이션에 **추가**할 때 그 사원의 사원번호는 반드시 기존 사원의 사원번호와 같지 않아야 한다는 제약조건은 제1정규형의 **원자성**과 관계있다.**(×)**
  → 주어진 내용은 **개체 무결성**과 관계가 있다.
  → 기본키(primary key) : Null을 허용하지 않고, 중복된 값도 허용하지 않는다.**(유일성)**
  → 제1정규형은 릴레이션의 모든 속성이 원자값으로 구성되어 있다.

// 트리거(trigger)와 주장(assertion)

- 먼저, 트리거와 주장은 데이터베이스 무결성 제약조건을 위해 사용할 수 있다.
- 트리거는 데이터베이스가 갱신될 때마다 DBMS가 자동으로 수행하는 프로시저이다.
- 트리거는 데이터베이스 무결성을 유지하기 위한 일반적이고 강력한 도구이다.
- 트리거는 제약조건을 위반했을 때 수행할 동작을 명시하는 것이다.
- 주장은 제약조건을 위반하는 연산은 수행되지 않도록 하는 것이다.
- 주장은 트리거보다 좀 더 일반적인 무결성 제약조건에 적용한다.
  → 두 개 이상의 테이블에 영향을 미치는 제약조건을 명시하기 위해 사용될 수 있다.

정답 : ③