



# SQL 활용

## 트리거

## 학습내용

- 무결성 규정
- 트리거 활용

## 학습목표

- 무결성의 의미 및 무결성 규정에 대하여 설명할 수 있다.
- 트리거의 구동 방법을 이해하고, 특정 사건에 대한 트리거를 작성할 수 있다.

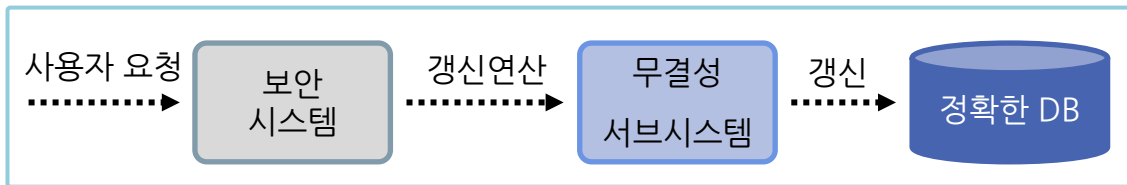
## ● 무결성 규정

### 1. 무결성의 의미

#### ◆ 무결성

- 정밀성, 정확성, 정당성
- 허가 받은 사용자가 수행하는 갱신 작업에서 의미적 오류를 방지함
- 의미적 제약의 개념

#### ◆ 무결성 서브 시스템



- 일관성
  - 데이터베이스에서 병행 트랜잭션들이 상호 작용으로부터 의미상 모순이 없도록 하는 것
- 신뢰성
  - 시스템의 오동작으로부터 오류가 발생하지 않도록 하는 것
- DBMS의 한 구성 요소
- 무결성 규정을 유지 관리함
- 데이터베이스의 무결성을 유지함
- 트랜잭션이 수행하는 갱신 연산이 무결성 규정을 위반하지 않는가를 감시함
  - 위반 시 거부, 보고, 취소 / 복귀를 수행함

## ● 무결성 규정

---

### 2. 제약 조건

#### ◆ 무결성 규정 대상

- ① 도메인
  - 형식
  - 타입
  - 범위
- ② 기본키, 외래키
  - 개체 무결성(Entity Integrity)
  - 참조 무결성(Referencial Integrity)
- ③ 종속성(묵시적 제약조건)
  - 함수 종속
  - 다치 종속
  - 조인 종속
- ④ 관계
  - 내부 관계
  - 외부 관계

#### ◆ 도메인 무결성 대상

- 도메인 정의
  - 도메인 이름
  - 데이터 형
- 삽입이나 갱신 연산에 적용

## ● 무결성 규정


---

### 2. 제약 조건

#### ◆ 릴레이션의 무결성 규정

- 릴레이션을 조작하는 과정에서의 의미적 제약조건을 명세함
- 연산 수행 전 / 후에 대한 제약 조건을 규정함
  - 삽입
  - 삭제
  - 갱신
- 분류
  - ① 상태 제약과 과도 제약
  - ② 집합 제약과 튜플 제약
  - ③ 즉시 제약과 지연 제약

#### ◆ 상태 제약과 과도 제약

- 상태 제약
    - 릴레이션 상태에 대한 제약
      - 일관성 있는 상태 유지
- 
- 정적 제약
    - 각 릴레이션 상태가 모두 만족해야 하는 규정
    - 데이터베이스 상태의 유효성

## ● 무결성 규정

### 2. 제약 조건

#### ◆ 상태 제약과 과도 제약

##### ● 상태 제약

- 키 속성의 제약 : 유일성
- NULL 값의 제약 : 이름은 NULL 값일 수 없음
- 관계 제약 : 참조 무결성

예 모든 학생은 하나의 학과에만 속함

- 도메인 제약 : 유효한 값

예 학생의 학년은 1, 2, 3, 4 중에 하나이어야 함

- 의미 무결성 제약

예 직원의 월급은 그의 관리자의 월급보다 높을 수 없음

##### ● 과도 제약

- 동적 제약
- 데이터베이스의 한 상태에서 다른 상태로 변환되는 과정에서 적용되는 규정
- 데이터베이스 상태의 변환 전과 후의 비교
  - 변환 전과 후에 모두 적용됨

예 월급은 감소될 수 없음

#### ◆ 집합 제약과 튜플 제약

##### ● 집합 제약

- 튜플 집합 전체에 대한 제약

예 직원 전체의 평균 급여는 300을 넘을 수 없음

##### ● 튜플 제약

- 처리되고 있는 튜플에만 적용됨

예 직원 급여의 최대는 500을 넘을 수 없음

⇒ 한 직원의 급여를 변경할 때 500이 넘는지만 감시하면 됨

## ● 무결성 규정

---

### 2. 제약 조건

#### ◆ 즉시 제약과 지연 제약

- 즉시 제약
  - 삽입 / 삭제 / 갱신 연산이 수행될 때마다 적용되는 제약 규정
- 지연 제약
  - 트랜잭션이 완전히 수행된 후에 적용되는 제약 규정

## ● 트리거 활용

### 1. 트리거의 개념

#### ◆ 트리거(TRIGGER)

- 방아쇠, 제동기, 계기, 유인, 자극이란 사전적 의미
- DBMS에서 특정 사건이 발생 시 자동으로 일련의 과정이 수행되는 프로시저

#### ◆ 프로시저 Vs. 트리거

- 프로시저 : 사용자가 직접 EXEC 명령어를 이용하여 프로시저를 수행함
- 트리거 : 특정 조건을 만족하면 자동으로 수행되도록 하는 저장 프로시저
  - ⇒ 특정 사건이 발생할 때만 실행되는 프로시저
  - ⇒ 사용자가 트리거를 따로 호출할 필요 없음

#### ◆ 무결성과 트리거(TRIGGER)

- 트리거는 데이터의 변경이 발생할 때 수행됨
  - ⇒ 데이터 변경 시 무결성에 문제가 발생되면 이를 보완할 수 있도록 자동으로 프로시저를 수행하도록 트리거를 정의해 놓으면 무결성을 유지시킬 수 있음
- 단점 : 테이블 선언 시 정의한 제약조건에 비하여 성능이 저하됨
- 장점 : 프로시저와 더불어 데이터베이스 내에 업무 규칙을 구현할 수 있음

#### ◆ 수행 기점에 따른 트리거의 분류

- AFTER 트리거
  - 이벤트(삽입 / 삭제 / 변경) 발생 직후 실행되는 트리거
  - 테이블에 대해서만 작성됨
- BEFORE 트리거
  - 이벤트(삽입 / 삭제 / 변경) 발생 이전에 실행되는 트리거
  - 일반적으로 BEFORE 트리거는 지원되지 않음
- INSTEAD OF 트리거
  - 이벤트(삽입 / 삭제 / 변경) 발생 시 해당 이벤트 대신 구동되는 트리거
    - ⇒ 다른 작업을 수행하는 트리거
  - INSTEAD OF 트리거를 활용하여 BEFORE 트리거 같은 역할을 수행시킬 수 있음



## ● 트리거 활용

### 1. 트리거의 개념

#### ◆ inserted와 deleted 테이블

- 트리거 작동 시 생성되는 임시 테이블
  - 사건에 따라서 둘 중 하나 또는 둘 다 만들어짐

| 사건 | inserted 테이블   | deleted 테이블    |
|----|----------------|----------------|
| 삽입 | 방금 삽입된 튜플이 복사됨 | -              |
| 변경 | 변경된 튜플이 복사됨    | 변경 전 튜플을 보관함   |
| 삭제 | -              | 방금 삭제된 튜플을 보관함 |

## ● 트리거 활용

### 2. 트리거의 구동

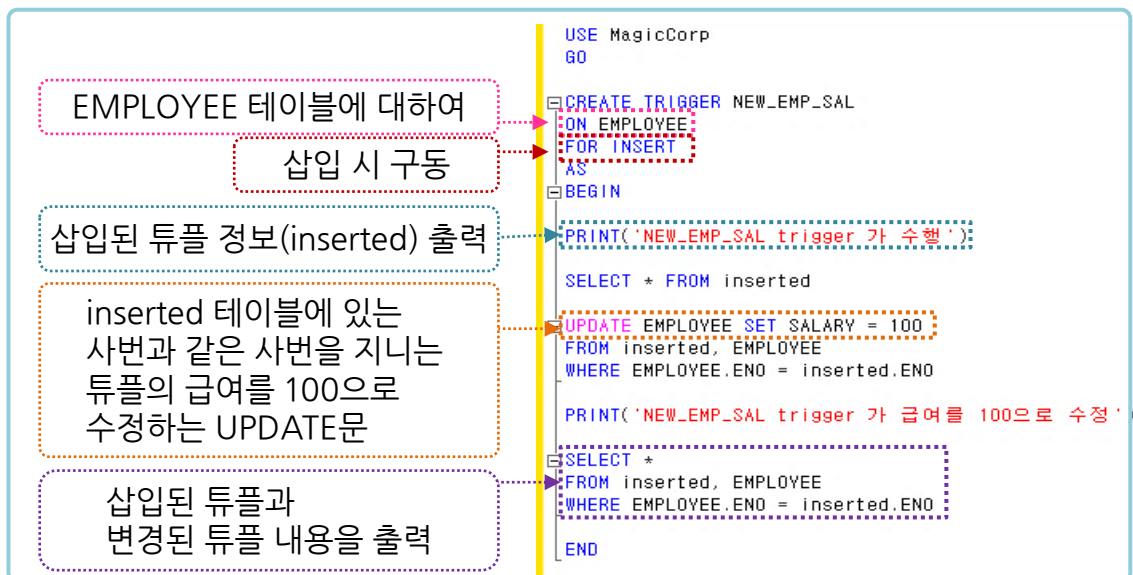
#### ◆ 트리거 생성 문법

```
CREATE TRIGGER 트리거명  
ON 테이블명  
[for / after / instead of] [insert / update / delete]  
AS  
SQL문
```

- for는 after와 같은 것임
- ON 테이블에 의해 테이블에 내용이 추가 / 삭제되면 inserted 또는 deleted라는 가상 테이블에 자동으로 추가되고 이를 이용하여 트리거를 수행시키게 됨

#### ◆ AFTER 트리거(또는 FOR 트리거) 구동 예

- 새로운 업무 규칙
  - 신입 직원들의 급여는 무조건 100임
  - 기존 직원들을 이 규칙에 적용 받지 않음
  - Employee 테이블에 새로운 튜플이 들어 올 때 마다 salary 속성을 자동으로 100이 되게 하면 됨



## ● 트리거 활용

### 2. 트리거의 구동

#### ◆ AFTER 트리거(또는 FOR 트리거) 구동 예

Q 사원 테이블이 300번인 사원 삽입하기

```
USE MagicCorp
GO
insert into EMPLOYEE(ENO, ENAME) values (300, 'new_emp')
```

100 %

결과 메시지

|   | ENO | ENAME   | JOB  | MANAGER | HIREDATE | SALARY | COMMISSION | DNO  |
|---|-----|---------|------|---------|----------|--------|------------|------|
| 1 | 300 | new_emp | NULL | NULL    | NULL     | NULL   | NULL       | NULL |

|   | ENO | ENAME   | JOB  | MANAGER | HIREDATE | SALARY | COMMISSION | DNO  | ENO | ENAME   | JOB  | MANAGER | HIREDATE | SALARY | COMMISSION |
|---|-----|---------|------|---------|----------|--------|------------|------|-----|---------|------|---------|----------|--------|------------|
| 1 | 300 | new_emp | NULL | NULL    | NULL     | NULL   | NULL       | NULL | 300 | new_emp | NULL | NULL    | NULL     | 100    | NULL       |

급여가 100으로 설정됨

결과 메시지

NEW\_EMP\_SAL trigger 가 수행

(1개 행이 영향을 받음)

(1개 행이 영향을 받음)

NEW\_EMP\_SAL trigger 가 급여를 100으로 수정

(1개 행이 영향을 받음)

(1개 행이 영향을 받음)

## ● 트리거 활용

### 2. 트리거의 구동

#### ◆ INSTEAD OF 트리거 구동 예

##### ● INSTEAD OF 트리거

- 뷰나 테이블에 삽입 / 삭제 / 변경 연산에 대응되어 다른 작업을 수행하는 트리거

##### ① DEPARTMENT 테이블에 갱신이 일어나면 갱신이 불가능하다고 메시지 출력하기

- 갱신은 일어나지 않게 하기

```
USE MagicCorp
GO

CREATE TRIGGER NO_UPDATE
ON DEPARTMENT
INSTEAD OF UPDATE
AS
BEGIN
    PRINT( 'DEPARTMENT에 대한 UPDATE는 하지 마시오' )
END
```

##### ② DEPARTMENT의 LOC 속성값을 모두 'SEOUL'로 변경하기

- 트리거로 인하여 안됨

```
USE MagicCorp
GO

SELECT * FROM DEPARTMENT

UPDATE DEPARTMENT
SET LOC = 'SEOUL'

SELECT * FROM DEPARTMENT
```

| 결과 |     | 메시지        |         |  |
|----|-----|------------|---------|--|
|    | DNO | DNAME      | LOC     |  |
| 1  | 10  | Accounting | Seoul   |  |
| 2  | 20  | Human      | Incheon |  |
| 3  | 30  | Sales      | Yungin  |  |
| 4  | 40  | Computing  | Suwon   |  |

|   | DNO | DNAME      | LOC     |  |
|---|-----|------------|---------|--|
| 1 | 10  | Accounting | Seoul   |  |
| 2 | 20  | Human      | Incheon |  |
| 3 | 30  | Sales      | Yungin  |  |
| 4 | 40  | Computing  | Suwon   |  |

| 결과 | 메시지   |
|----|---|
|    | (4개 행이 영향을 받음)<br>DEPARTMENT에 대한 UPDATE는 하지 마시오 |
|    | (4개 행이 영향을 받음)                                  |
|    | (4개 행이 영향을 받음)                                  |

## ● 트리거 활용

### 3. DDL 트리거

#### ◆ DDL 트리거

- CREATE, ALTER, DROP과 같은 DDL문이 발생시 구동되는 트리거
  - DML 트리거와 유사함
  - 정의 시 ON 테이블명 대신 ON DATABASE를 사용함
  - INSTEAD OF 트리거는 지원 안 함

```
CREATE TRIGGER 트리거명
ON DATABASE
{FOR|AFTER} {DROP_TABLE|CREATE_TABLE|ALTER_TABLE}
AS SQL문
```

**Q** 테이블 삭제 시 자동으로 ROLLBACK이 발생되어 테이블이 삭제되지 않도록 하는 DDL 트리거 만들기

```
USE MagicCorp
GO

CREATE TRIGGER ROLLBACK_TRIGGER
ON DATABASE
FOR DROP_TABLE
AS
BEGIN
PRINT('DDL Tigger: ROLLBACK')

ROLLBACK TRANSACTION
END
```

- EMPLOYEE 테이블을 삭제해도 삭제되지 않음

```
USE MagicCorp
GO

DROP TABLE EMPLOYEE
GO

SELECT * FROM EMPLOYEE
GO
```

100 % <

**결과** **메시지**

DDL Trigger: ROLLBACK  
메시지 3609, 수준 16, 상태 2, 줄 4  
트리거가 발생하여 트랜잭션이 종료되었습니다. 일괄 처리가 중단되었습니다.  
(15개 행이 영향을 받음)

## ● 트리거 활용

---

### 3. DDL 트리거

#### ◆ 트리거의 변경과 삭제

- DCL문임으로 DROP과 ALTER문을 씀

```
DROP TRIGGER 트리거명
```

```
ALTER TRIGGER 트리거명
```

# 핵심요약

## 1. 무결성 규정

### ■ 무결성의 의미

#### ■ 무결성

- 정밀성, 정확성, 정당성
- 허가 받은 사용자가 수행하는 갱신 작업에서 의미적 오류를 방지함
- 의미적 제약의 개념

#### ■ 무결성 서브 시스템

- 사용자 요청 ⇨ 보안 시스템 ⇨ 갱신연산 ⇨ 무결성 서브시스템 ⇨ 갱신 ⇨ 정확한 DB
- 무결성 규정을 유지 관리함
- 데이터베이스의 무결성을 유지함
- 트랜잭션이 수행하는 갱신 연산이 무결성 규정을 위반하지 않는가를 감시함
  - ▶ 위반 시에는 거부, 보고, 취소 / 복귀를 수행함

# 핵심요약

## 1. 무결성 규정

### ■ 제약 조건

#### ■ 무결성 규정 대상

- 도메인 : 형식, 타입, 범위
- 기본키, 외래키 : 개체 무결성(Entity Integrity), 참조 무결성(Referencial Integrity)
- 종속성 (목시적 제약조건) : 함수 종속, 다치 종속, 조인 종속
- 관계 : 내부 관계, 외부 관계

#### ■ 도메인 무결성 대상

- 도메인 정의 : 도메인 이름, 데이터 형
- 삽입이나 갱신 연산에 적용

#### ■ 릴레이션의 무결성 규정

- 릴레이션을 조작하는 과정에서의 의미적 제약조건을 명시함
- 연산 수행 전 / 후에 대한 제약 조건을 규정함
  - ▶ 삽입
  - ▶ 삭제
  - ▶ 갱신
- 분류
  - ▶ 상태 제약과 과도 제약
  - ▶ 집합 제약과 튜플 제약
  - ▶ 즉시 제약과 지연 제약



# 핵심요약

## 1. 무결성 규정

### ■ 제약 조건

#### ■ 상태 제약

- 릴레이션 상태에 대한 제약
- 일관성 있는 상태 유지 : 정적 제약
  - ▶ 각 릴레이션 상태가 모두 만족해야 하는 규정
  - ▶ 데이터베이스 상태의 유효성

- 키 속성의 제약 : 유일성
- NULL 값의 제약 : 이름은 NULL 값일 수 없음
- 관계 제약 : 참조무결성
- 도메인 제약 : 유효한 값
- 의미 무결성 제약

#### ■ 과도 제약

- 동적 제약
- 데이터베이스의 한 상태에서 다른 상태로 변환되는 과정에서 적용되는 규정
- 데이터베이스 상태의 변환 전과 후의 비교
  - ▶ 변환 전과 후에 모두 적용됨
  - 예) 월급은 감소될 수 없음

## 핵심요약

### 1. 무결성 규정

#### ■ 제약 조건

- 집합 제약
  - 튜플 집합 전체에 대한 제약
- 튜플 제약
  - 처리되고 있는 튜플에만 적용됨
- 즉시 제약
  - 삽입 / 삭제 / 갱신 연산이 수행될 때마다 적용되는 제약 규정
- 지연 제약
  - 트랜잭션이 완전히 수행된 후에 적용되는 제약 규정

# 핵심요약

## 2. 트리거 활용

### ■ 트리거의 개념

- DBMS에서 특정 사건이 발생 시 자동으로 일련의 과정이 수행되는 프로시저

- 프로시저

- ▶ 사용자가 직접 EXEC 명령어를 이용하여 프로시저를 수행함

- 트리거

- ▶ 특정 조건을 만족하면 자동으로 수행되도록 하는 저장 프로시저
  - ▶ 특정 사건이 발생될 때만 실행되는 프로시저
  - ▶ 사용자가 트리거를 따로 호출할 필요 없음

### ■ 무결성과 트리거(TRIGGER)

- 트리거는 데이터의 변경이 발생할 때 수행됨

- ▶ 데이터 변경 시 무결성에 문제가 발생되면 이를 보완할 수 있도록 자동으로 프로시저를 수행하도록 트리거를 정의해 놓으면 무결성을 유지시킬 수 있음

- 단점

- ▶ 릴레이션 선언 시 정의한 제약조건에 비하여 성능이 저하됨

- 장점

- ▶ 프로시저와 더불어 데이터베이스 내에 업무 규칙을 구현할 수 있음

# 핵심요약

## 2. 트리거 활용

### ■ 트리거의 개념

#### ■ 수행 기점에 따른 트리거의 분류

##### - AFTER 트리거

- ▶ 이벤트(삽입 / 삭제 / 변경) 발생 직후 실행되는 트리거
- ▶ 테이블에 대해서만 작성됨

##### - BEFORE 트리거

- ▶ 이벤트(삽입 / 삭제 / 변경) 발생 이전에 실행되는 트리거
- ▶ 일반적으로 BEFORE 트리거는 지원되지 않음

##### - INSTEAD OF 트리거

- ▶ 이벤트(삽입 / 삭제 / 변경) 발생 시 해당 이벤트 대신 구동되는 트리거
- ▶ 즉, 다른 작업을 수행하는 트리거
- ▶ INSTEAD OF 트리거를 활용하여 BEFORE 트리거 같은 역할을 수행시킬 수 있음

#### ■ inserted와 deleted 테이블

| 사건 | inserted 테이블   | deleted 테이블    |
|----|----------------|----------------|
| 삽입 | 방금 삽입된 튜플이 복사됨 | -              |
| 변경 | 변경된 튜플이 복사됨    | 변경 전 튜플을 보관함   |
| 삭제 | -              | 방금 삭제된 튜플을 보관함 |

# 핵심요약

## 2. 트리거 활용

### ■ 트리거의 구동

#### ■ 트리거 생성 문법

```
CREATE TRIGGER 트리거명  
ON 테이블명  
[for / after / instead of]  
[insert / update / delete]  
AS  
SQL문
```

- for는 after와 같은 것임
- on 테이블에 의해 테이블에 내용이 추가 / 삭제되면 inserted 또는 deleted라는 가상 테이블에 자동으로 추가되고 이를 이용하여 트리거를 수행시키게 됨

#### ■ AFTER 트리거(또는 FOR 트리거) 구동 예

##### - 새로운 업무 규칙

- ▶ 신입 직원들의 급여는 무조건 100임
- ▶ 기존 직원들을 이 규칙에 적용 받지 않음
- ▶ Employee 테이블에 새로운 튜플이 들어 올 때 마다 salary속성을 자동으로 100이 되게 하면 됨

##### - INSTEAD OF 트리거

- ▶ 뷰나 테이블에 삽입 / 삭제 / 변경 연산에 대응되어 다른 작업을 수행하는 트리거

## 핵심요약

### 2. 트리거 활용

#### ■ 트리거의 구동

##### ■ DDL 트리거

- CREATE, ALTER, DROP과 같은 DDL문이 발생시 구동되는 트리거
- 정의 시 ON 테이블명 대신 ON DATABASE를 사용함
- INSTEAD OF 트리거는 지원 안 함

```
CREATE TRIGGER 트리거명  
ON DATABASE  
{FOR|AFTER} {DROP_TABLE|CREATE_TABLE|ALTER_TABLE}  
AS SQL문
```

- 트리거의 변경과 삭제

- ▶ DCL문임으로 DROP과 ALTER문을 씀

```
DROP TRIGGER 트리거명
```

```
ALTER TRIGGER 트리거명
```