Chap 5. Advanced SQL
[Supplementary Materials]

Revision by Gun-Woo Kim

Dept. of Computer Science and Engineering

Hanyang University

Database System Concepts, 6th Ed.

Triggers (5.15)

PL/SQL의 PL:Program Language

트리거 (trigger)

- 특정 릴레이션에 삽입과 같은 어떤 **사건**들이 **조건**에 의해 구동되어, 적절한 **조치**를 할 수 있는 능동적 요소

사건-조건-조치 (Event-Condition-Action) 규칙

- 또는 ***ECA*** rule 이라고 불린다.

```
INSERT INTO Book VALUES(14, '스포츠 과학 1', '이상미디어', 25000);

SELECT * FROM Book WHERE bookid='14';
SELECT * FROM Book_log WHERE bookid_l='14'; /* 결과 확인 */
```

```
1개 행 이(가) 삽입되었습니다.
삽입 투플을 Book_log 테이블에 백업..
```

Book 테이블에 튜플을 삽입하여 트리거가 실행된 결과 (자동으로 Book_log에 튜플 삽입)

BOOKID	BOOKNAME	PUBLISHER	PRICE
14	스포츠 과학 1	이상미디어	25000

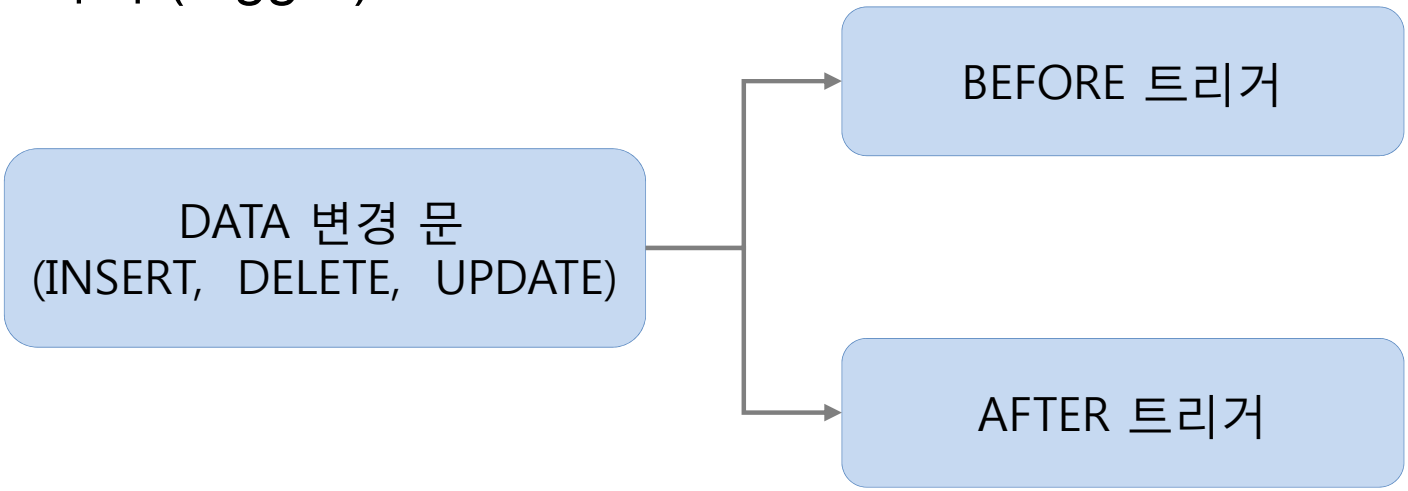
<Book>

BOOKID_L	BOOKNAME_L	PUBLISHER_L	PRICE_L
14	스포츠 과학 1	이상미디어	25000

<Book_log>

Triggers (5.15)

트리거 (trigger)



Before, After 트리거 존재

데이터의 변경(INSERT, UPDATE, DELETE)이 일어날 때 부수적으로 필요한 '데이터의 기본값 제공', '데이터 제약 준수', 'SQL 뷰의 수정', '참조 무결성 작업'등을 수행

Triggers (5.16)

E.g. *time_slot_id* is not a primary key of *timeslot,* so we cannot create a foreign key constraint from *section* to *timeslot.*

ORACLE Version

create or replace trigger *timeslot_check1* **after insert on** *section*
referencing new as *nrow*
for each row 표준에서는 new as 가 아니라 new row as임
declare
temp_time_slot_id integer; / when절에 Subquery 지원 안함*/*
begin
select count(time_slot_id) into temp_time_slot_id
from timeslot
where time_slot_id = :nrow.time_slot_id;
if *temp_time_slot_id <1*
then
RAISE_APPLICATION_ERROR(-20007. '참조무결성 오류');
end if; */* Rollback 지원 안함*
end;

Triggers (5.19)

```
create or replace trigger credits_earned after update on
takes
referencing old as orow new as nrow
for each row
when (((nrow.grade <> 'F') and (nrow.grade is not null))
   and ((orow.grade = 'F') or (orow.grade is null)))
begin /*atomic 구문 안됨*/
    update student
    set tot_cred= tot_cred +
        (select credits
         from course
         where course.course_id= :nrow.course_id)
    where student.id = :nrow.id;
end;
```

Triggers (정리)

Before와 After

- Before: When 조건이 사건 이전에 검사
- After: When 조건이 사건 이후에 검사

사건의 종류

- Insert, Delete, Update
  - OF – Attribute 절
    - → Update에 사용가능 (Insert, Delete에는 사용되지 않음)
- When 절
  - 생략될 경우, 사건이 발생하면 항상 조치가 실행

OLD ROW AS와 NEW ROW AS

- 갱신 이전 튜플과 갱신 이후 튜플
  - OLD ROW AS나 NEW ROW AS절을 이용하여 이름 부여
- 사건이 Insert인 경우, OLD ROW AS는 허용되지 않음
  - 사건이 delete인 경우, NEW ROW AS는 허용되지 않음

Ranking (5.21)

Find the rank of each student.

select *ID*, **rank**() **over** (**order by** *GPA* **desc**) **as** *s_rank*
from *student_grades*

select *ID*, **dense_rank**() **over** (**order by** *GPA* **desc**) **as** *s_rank*
from *student_grades*

student_grades

ID	GPA
12345	80
19991	42
23121	65
54321	65
67890	50

Rank

ID	s_rank
12345	1
23121	2
54321	2
67890	4
19991	5

Dense_Rank

ID	s_rank
12345	1
23121	2
54321	2
67890	3
19991	4

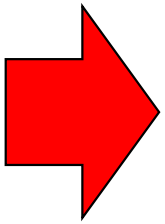
Ranking (5.23)

"Find the rank of students within each department."

```
select ID, dept_name,
       rank () over (partition by dept_name order by GPA desc)
              as dept_rank
from dept_grades
order by dept_name, dept_rank;
```

dept_grades

ID	dept_name	GPA
12345	Biology	80
19991	Biology	42
23121	Comp. Sci.	65
54321	Comp. Sci.	65
67890	Comp. Sci.	50
47893	Elec. Eng.	70



Partition by Ranking

ID	dept_name	dept_rank
12345	Biology	1
19991	Biology	2
23121	Comp. Sci.	1
54321	Comp. Sci.	1
67890	Comp. Sci.	3
47893	Elec. Eng.	1

Ranking (5.23)

Can be used to find top-n results

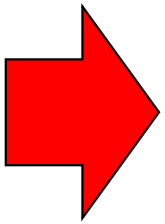
More general than the **limit** *n* clause supported by many databases, since it allows top-n within each partition

ORACLE Version

```
select * from
    (select ID, rank () over (order by GPA desc)
                as s_rank
     from student_grades)
where s_rank <=3;
```

student_grades

ID	GPA
12345	80
19991	42
23121	65
54321	65
67890	50



ID	s_rank
12345	1
23121	2
54321	2

Ranking (5.25)

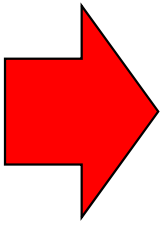
For a given constant *n*, the ranking the function *ntile*(*n*) takes the tuples in each partition in the specified order, and divides them into *n* buckets with equal numbers of tuples.

E.g.,

select *ID*, **ntile**(4) **over** (**order by** *GPA* **desc**) **as** *quartile*
from *student_grades;*

student_grades

ID	GPA
12345	80
19991	42
23121	65
54321	65
67890	50
47893	70



ID	Quartile
12345	1
47893	1
21121	2
54321	2
67890	3
19991	4

Windowing (5.26)

Window specification in SQL: 부분적으로 집계함수 쓸때 사용

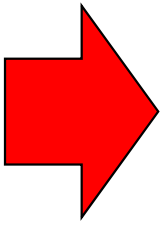
Given relation *sales(date, value)*

select *date*, ***sum***(*value*) **over**
(**order by** *date* **between rows** 1 **preceding and** 1 **following**)
from *sales*

자기자신 이전 자기자신 이후

sales

Date	Value
17/11/16	2000
17/11/15	1500
17/11/11	900
17/11/12	1000
17/11/07	1300



Date	Sum(value)
17/11/07	2200
17/11/11	3200
17/11/12	3400
17/11/15	4500
17/11/16	3500