

Day30_20220810 insert, update, delete, commit

| | |
|-------|--------------------------|
| 강의 번호 | |
| 유형 | |
| 자료 | |
| 복습 | <input type="checkbox"/> |
| 속성 | |
| 작성일자 | @2022년 8월 10일 오후 5:42 |

Oracle

From 절 서브 쿼리

From 절에서 서브쿼리를 사용하면 특정 조건식을 갖는 SELECT 문을 테이블처럼 사용할 수 있다. 마치 가상의 테이블(View)과 같은 역할을 할 수 있어 인라인 뷰(Inline View)라고도 부른다.

SELECT 열이름

FROM 테이블이름 AS 별칭1,

(SELECT 열이름

FROM 테이블이름

WHERE 조건식) AS 별칭2

WHERE 별칭1.열이름 = 별칭2.열이름2

Quiz 1

직원 employees 테이블 중에서 department_name 이 IT인 직원의 정보를 인라인 뷰를 이용하여 출력하세요 (employees, departments 2개의 테이블을 이용하세요)

| | |
|---|-------------------------------|
| 1 | SELECT department_id |
| 2 | FROM hr.departments |
| 3 | WHERE department_name = 'IT'; |
| 4 | |

| DEPARTMENT_ID |
|---------------|
| 60 |

```
SELECT *
FROM hr.employees e,
     (SELECT department_id
      FROM hr.departments
      WHERE department_name = 'IT') d
WHERE e.department_id = d.department_id;
```

SQL Worksheet

Clear

Find

Actions

Save

Run

```
1 SELECT *
2 FROM hr.employees e,
3     (SELECT department_id
4      FROM hr.departments
5      WHERE department_name = 'IT') d
6 WHERE e.department_id = d.department_id;
```

| LAST_NAME | EMAIL | PHONE_NUMBER | HIRE_DATE | JOB_ID | SALARY | COMMISSION_PCT | MANAGER_ID | DEPARTMENT_ID | DEPARTMENT_ID |
|-----------|----------|--------------|-----------|---------|--------|----------------|------------|---------------|---------------|
| Hunold | AHUNOLD | 590.423.4567 | 03-JAN-06 | IT_PROG | 9000 | - | 102 | 60 | 60 |
| Ernst | BERNST | 590.423.4568 | 21-MAY-07 | IT_PROG | 6000 | - | 103 | 60 | 60 |
| Austin | DAUSTIN | 590.423.4569 | 25-JUN-05 | IT_PROG | 4800 | - | 103 | 60 | 60 |
| Pataballa | VPATABAL | 590.423.4560 | 05-FEB-06 | IT_PROG | 4800 | - | 103 | 60 | 60 |
| Lorentz | DLORENTZ | 590.423.5567 | 07-FEB-07 | IT_PROG | 4200 | - | 103 | 60 | 60 |

서브 쿼리 만드는 팁 !!!

- 들어쓰기를 한다.
- 분리하여 실행한다.
- 주석을 사용하자

DML Data Manipulation Language 데이터 조작어

SELECT 구문도 DML에 속하기는 하지만 단지 조회하여 출력할 뿐이다.

하지만 지금부터 살펴볼 DML의 INSERT, UPDATE, DELETE은 데이터의 원본을 직접 삽입, 갱신, 삭제하는 명령어들이다. 이렇게 직접 데이터를 조작하여 저장까지 하는 일련의 과정을 트랜잭션 Transaction이라고 한다.

INSERT 행 삽입

INSERT 명령어를 통해서 새로운 데이터를 테이블에 해단위로 직접 삽입하게 된다.

INSERT INTO 테이블 이름 (열이름1, 열이름2, 열이름3.....)

VALUES (값1, 값2, 값3.....)

- 모든 열 즉 전체열에 데이터를 전부 넣을 때에는 열이름들의 나열은 생략 가능하다.

Quiz 3

departments 테이블에 id가 271 name이 'Sample_Dept' manager_id가 200 location_id가 1700인 행을 넣으세요.

- 열이름과 입력되는 데이터 값이 일치하는지 확인하기 위해 열이름은 생략가능하지만 모두 기술할것을 권장한다. 입력할때 데이터 타입과 순서등이 정확히 일치해야 한다.

```
SELECT hr.departments(department_id, department_name, manager_id, location_id)
VALUES (271, 'Sample_Dept', 200, 1700);
```

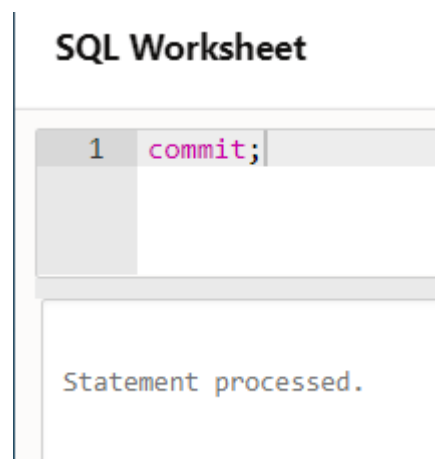
선생님이 기억이 안 나서 새로 작업해봄.

```
INSERT INTO address(zip_code, address_detail)
VALUES (90215, '서울');
```

```
SELECT *
FROM address;
```

- INSERT 명령어를 실행했다고 해서 데이터베이스에 즉시 영구적으로 반영되는 것은 아닙니다. 실행한 DML 명령어를 최종적으로 데이터베이스에 반영하려면 커밋 commit(영구저장)을 해야한다.

커밋을 실행하기 전까지 실행한 모든 DML 은 데이터베이스에 반영되지 않는다.



- 만약 커밋을 하지 않고 데이터베이스를 비정상적으로 종료하게 되면 커밋이전에 실행한 모든 내용은 반영되지 않고 돌아간다.

UPDATE

update는 기존의 저장된 데이터 값을 다른 데이터 값으로 변경할 때 사용한다 . update절에는 where 조건절을 사용할 수 있다. 따라서 갱신 하려는 대상의 조건을 적용하여 대량의 데이터도 한번에 수정이 가능하다.

UPDATE 테이블이름

SET 열이름 = 데이터값

WHERE 조건식;

Quiz 3

address 테이블에서 zip_code가 90215인 행을 찾아서 address_detail 값을 '인천'으로 변경하세요.

DELETE 행 삭제

DELETE 명령어를 사용하여 테이블의 데이터를 삭제할 수 있다. UPDATE와 마찬가지로 WHERE절을 사용하여 조건을 지정한다.

WHERE 절을 생략할 수도 있지만 생략하게 되면 특정 테이블의 데이터가 한번에 전부 즉시 삭제되므로 주의가 필요하다.

DELETE 테이블 이름

[WHERE 조건식];

Quiz 3

address 테이블에서 address_detail 이 인천인 데이터를 전부 삭제하세요.

```
delete address
WHERE address_detail = '인천';

commit;
```

```
1 DELETE address
2 WHERE address_detail = '인천';
3
4 commit;
```



A.C.I.D. - 관계형 데이터베이스 트랜잭션의 가장 기본적인 원리이자 특징

| | | |
|-----|--|-------------|
| 원자성 | 트랜잭션의 처리가 완전히 끝나지 않았을 경우에는 전혀 이루어지지 않은 것과 같아야 한다. | Atomicity |
| 일관성 | 트랜잭션의 처리가 성공적으로 완료되면 데이터베이스는 모순 없이 일관성이 보존된 상태여야 한다. | Consistence |
| 고립성 | 어떤 트랜잭션도 다른 트랜잭션의 부분 실행 결과를 볼 수 없다. | Isolation |

| | | |
|-----|--------------------------------------|------------|
| 지속성 | 트랜잭션이 성공하면 트랜잭션의 결과는 영구적으로 보장되어야 한다. | Durability |
|-----|--------------------------------------|------------|

원자성 - 예를 들면 은행에서 10만원 이체를 시도하는 중에 문제가 발생한다면 전혀 이체가 이루어지지 않아야지 5만원만 이체가 이루어진다고 7만원만 이체가 발생하는 경우는 없어야 한다.

all or nothing

일관성 - 트랜잭션이 완료되면 이체가 이상없이 성공하면 그 데이터는 일관되게 유지되어야만 한다. 예를 들면 A계좌에서 10만원이 출금되면 B계좌에는 10만원이 반드시 입금되어야만 한다.

고립성 - 트랜잭션이 완료되지 않은 동안에는 다른 트랜잭션이 참조하거나 변경할 수 없다. 예를 들면 '가'라는 사용자의 계좌에서 이체가 실행되는 것에 대해 '나'사용자는 관여할 수 없다. '가'사용자의 이체 실행이 완료되지 않으면 '나'사용자는 계좌를 조회할 수 있을뿐 다른 계좌로 이체 처리를 할 수 없어야 한다.

지속성 - 트랜잭션이 정상적으로 완료되면 해당 데이터는 저장되어 보존되어야 한다. 보존이 보장됨으로 데이터베이스 전체 시스템에 대한 신뢰성과 일관성을 유지할 수 있다.

DDL : Data Definition Language 데이터 정의어

DDL 따로 커밋하지 않아도 즉시 데이터 베이스에 적용된다.

데이터베이스 생성, 데이터베이스 삭제, 테이블 생성, 테이블 삭제 등 데이터베이스 기본 생성에 관련된 것들로 주로 원타임 실행이 이루어진다.

CREATE 테이블 - 새로운 테이블을 생성할 때 사용된다.

CREATE TABLE 테이블 이름

(열이름1 데이터타입,

열이름2 데이터타입,

```
.  
.   
.   
);
```

sample_product라는 이름의 테이블을 pid(숫자), pname(글자20), pdate(날짜타입)

```
CREATE TABLE sample_product (  
  pid NUMBER(10),  
  pname VARCHAR2(20),  
  pdate DATE  
);
```

1 Television 22/08/10

```
INSERT INTO sample_product  
VALUES(1, 'Television', to_date('220810','YYMMDD'));
```

```
1 INSERT INTO sample_product  
2 VALUES(1, 'Television', to_date('220810','YYMMDD'));
```

```
1 row(s) inserted.
```

테이블 수정 ALTER

ALTER TABLE 테이블 이름

MODIFY (열이름1 데이터타입

열이름2 데이터타입

.
.
.
);

Quiz 5

sample_product 테이블에 있는 PNAME열의 데이터 타입과 크기를 char 타입의 15자리로 변경하세요.

```
ALTER TABLE sample_product  
modify(pname char(15));
```

```
1 ALTER TABLE sample_product  
2 modify(pname char(15));|
```

Table altered.

열 이름 수정

ALTER TABLE 테이블이름 RENAME COLUMN 열이름1 TO 열이름2;

Quiz 6

sample_product 테이블에 있는 PNAME열의 이름을 productname으로 변경하세요.

```
ALTER TABLE sample_product RENAME COLUMN pname TO productname;
```

```
1 ALTER TABLE sample_product RENAME COLUMN pname TO productname;
```

Table altered.

```
1 SELECT *
2 FROM sample_product;
```

| PID | PRODUCTNAME | PDATE |
|-----|-------------|-----------|
| 1 | Television | 10-AUG-22 |
| 1 | Television | 10-AUG-22 |

[Download CSV](#)

2 rows selected.

열 삭제

ALTER TABLE 테이블이름 DROP COLUMN 열이름;

Quiz 7

sample_product 테이블에 있는 productname으로 삭제하세요.

```
1 ALTER TABLE sample_product DROP COLUMN productname;|
```

Table altered.

```
1 SELECT *
2 FROM sample_product;|
```

| PID | PDATE |
|-----|-----------|
| 1 | 10-AUG-22 |
| 1 | 10-AUG-22 |

[Download CSV](#)
2 rows selected.

```
ALTER TABLE sample_product DROP COLUMN productname;
```

테이블 데이터만 전체 삭제 - 테이블에서 구조는 남기고 데이터만 모두 삭제.

TRUNCATE TABLE 테이블이름;

```
1 TRUNCATE TABLE sample_product;
```

Table truncated.

```
1 SELECT *  
2 FROM sample_product;
```

no data found

테이블 완전히 삭제 - 데이터와 구조를 모두 완전히 삭제

DROP TABLE 명령어는 테이블을 삭제할 때 모든 자료와 구조까지 삭제하고 사용하던 메모리까지 돌려준다. DDL 명령어이기 때문에 자동 커밋된다.

```
1 drop table sample_product;
```

Table dropped.

삭제 명령어들 비교

| | | |
|----------|-------------------------------|----------------|
| DELETE | DML | 데이터만 삭제 |
| TRUNCATE | DDL(Data Definition Language) | 구조는 남겨두고 모두 삭제 |
| DROP | DDL | 구조 포함 전체 삭제 |

뷰 - 가상의 테이블

테이블과 아주 유사하지만 실제 데이터는 없는 가상의 테이블이 뷰 view이다.

뷰를 사용하는 이유는 사용자의 편의와 보안 때문이다. 예를 들면 백화점에서 VIP회원들만 따로 실제 테이블에서 분리하여 가상의 뷰로 만들어 담당자를 지정하여 접근할 수 있게 한다면 그 직원은 전체 회원의 정보는 볼 수 없지만 뷰를 통해 VIP 회원들의 정보에만 접근하여 다양한 서비스와 프로모션 등을 제공할 수 있도록 할 수 있다.

임시로 복잡한 SQL문을 매번 작성하지 않아도 뷰를 한번 만들어놓고 사용자를 지정하면 쉽고 편리하게 사용할 수 있다.

- 뷰는 마치 실제 테이블처럼 사용할 수 있다.
- 뷰는 자주 쓰는 SQL 구문의 결과를 미리 만들어 놓는 개념이다.
- 여러 개의 테이블을 조인하여 하나의 뷰로 만들 수 있다.
- 사용자별로 접근 권한을 다르게 설정하여 보안 문제를 해결할 수 있다.

```

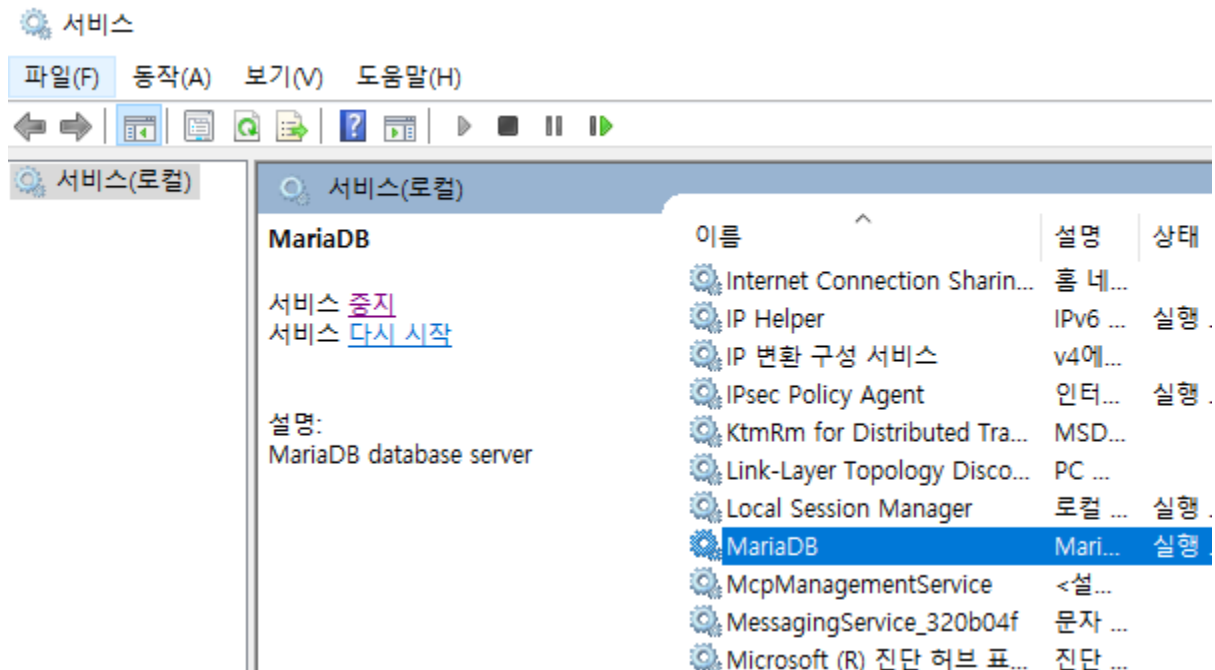
1 select *
2 from HR."EMP_DETAILS_VIEW";

```

| EMPLOYEE_ID | JOB_ID | MANAGER_ID | DEPARTMENT_ID | LOCATION_ID | COUNTRY_ID | FIRST_NAME | LAST_NAME |
|-------------|---------|------------|---------------|-------------|------------|------------|-----------|
| 100 | AD_PRES | - | 90 | 1700 | US | Steven | King |
| 101 | AD_VP | 100 | 90 | 1700 | US | Neena | Kochhar |
| 102 | AD_VP | 100 | 90 | 1700 | US | Lex | De Haan |
| 103 | IT_PROG | 102 | 60 | 1400 | US | Alexander | Hunold |
| 104 | IT_PROG | 103 | 60 | 1400 | US | Bruce | Ernst |
| 105 | IT_PROG | 103 | 60 | 1400 | US | David | Austin |
| 106 | IT_PROG | 103 | 60 | 1400 | US | Valli | Pataballa |

헤이디 접속 안 되는 문제.

중지 누른다.



시작을 누른다.

