Day30

Date	@10/08/2022
작성일시	@2022년 8월 10일 오전 9:39
강의 번호	BD101
유형	Data_SQL
강사명	AustinYoon
강의자료	☐
노션 복습	☐
코딩 복습	☐
주말숙제(교제)	☐
정리	☐

Day30

Aug

https://github.com/Kray273/SQLLab

FROM절 서브쿼리

👉 FROM절에서 서브쿼리를 사용하면 특정 조건식을 갖는 SELECT문을 테이블처럼 사용할 수 있다.
마치 가사의 테이블(VIEW)과 같은 역할을 할 수 있어 인라인 뷰(INLINE VIEW) 라고도 부른다.

```
SELECT 열이름1
FROM 테이블이름 AS 별칭1(
                      SELECT 열이름2
                      FROM 테이블 이름
                      WHERE 조건식
                      ) AS 별칭2
WHERE 별칭1.열이름1 = 별칭2.열이름2;
```

Quiz1

employees 테이블 중에서 department_name이 IT인 직원의 정보를 인라인 뷰를 이용하여 출력하라.

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

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_D
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY
105	David	Austin	DAUSTIN	590.423.4569	25-JUN
106	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB

👉 서브쿼리 만드는 팁!!
1. 들여쓰기를 한다.
2. 분리해서 실행한다.
3. 주석을 사용한다.

●DML_Data Manipulation Language

👉 데이터 조작어
SELECT구분도 DML에 속하기는 하지만 단지 조회 출력할 뿐이다.
하지만 지금부터 살펴볼 DML의 INSERT, UPDATE,DELETE는 데이터 원본을 수정하는 명령어이다.
이렇게 직접 데이터를 조작하여 저장까지 하는 일련의 과정을 Transaction이라고 한다.

1. INSERT ⇒ 행 삽입.

👉 INSERT 명령어를 통해서 새로운 데이터를 테이블에 행단위로 직접 삽입.

```
INSERT INTO 테이블 이름(열이름1, 열이름2...)
VELUES (값1, 값2.....)
/* 모든 열 즉 전체열에 데이터를 전부 넣을 때에는 열이름들의 나열은 생략 */
```

Quiz2

새로운 테이블에 데이터 삽입

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

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

👉 INSERT 명렁어를 실행했다고 해서 데이터베이스에 즉시 영구적으로 반영되는 것은 아니다.
실행한 DML명령어를 최종적으로 데이터베이스에 반영하려면 ‘commit’을 항상 해야한다.
커밋을 실행하기 전까지 실행한 모든 DML은 데이터베이스에 반영되지 않는다.

```
COMMIT;
```

```
Statement processed.
```

👉 만약 커밋을 하지 않고 데이터베이스를 비정상적으로 종료하게 되면 커밋이 전에 실행한 모든 내용은 반영되지 않는다.

2. UPDATE

👉 UPDATE은 기존의 저장된 데이터 값을 다른 데이터 값으로 변경할때 사용된다.
UPDATE절에는 WHERE 조건절을 사용할 수 있으므로 갱신하려는 대상의 조건을 적용하여
대량의 데이터를 한번에 수정할 수 있다.

```
UPDATE 테이블이름
SET 열이름 = 데이블값
WHERE 조건식
```

Quiz3

address 테이블에서 zip_code 가 90215 인 행을 찾아서 address_detail 값을 '인천'으로

```
UPDATE address
SET address_detail = '인천'
WHERE zip_code = 90215;
commit;
```

```
1 row(s) updated.
```

3. DELETE 행 삭제

👉 DELETE 명령어를 사용하여 테이블의 데이터를 삭제할 수 있다. UPDATE와 마찬가지로 WHERE절을 사용하여 조건을 지정한다.
WHERE절을 생략할 수도 있지만 생략하게 되면 특정 테이블의 데이터가 한번에 전부 즉시 삭제되므로 주의가 필요하다.

```
DELETE 테이블 이름
[WHERE 조건절];
```

Quiz4

address 테이블에서 address_detail 이 인천인 데이터를 전부 삭제하세요

```
DELETE address
WHERE  address_detail = '인천';
commit;
```

```
1 row(s) deleted.

Statement processed.
```

▼ A.C.I.D. - 관계형 데이터베이스 트랜잭션의 가장 기본적인 원리, 특징

▼ 원자성 - 트랜잭션의 처리가 완전히 끝나지 않았을 경우에는 전혀 이루어지지 않은 것과 같아야한다. Atomicity

👉 원자성 - 예를 들면 은행에서 10만원 이체를 시도하는 중에 문제가 발생한다면 전혀 이체가 이루어지지 않아야지 5만원만 이체가 이루어진다는가 7만원만이체가 발행되는 경우는 없어야 한다. all or nothing

▼ 일관성 - 트랜잭션의 처리가 성공적으로 완료되면 데이터베이스는 모순 없이 일관성이 보존된 상태여야 한다. Consistence

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

▼ 고립성 - 어떤 트랜잭션도 다른 트랜잰션의 부분 실행 결과를 볼수 없다. Isolation

👉 고립성 - 트랜잭션이 완료되지 않은 동안에는 다른 트랜잭션이 참조하거나 변경할수 없다. 예를 들면 '가'라는 사용자의 계좌에서 이체가 실행되는 것에 대해 '나'사용자는 관여할수 없다. '가'사용자의 이체 실행이 완료되지 않으면 '나' 사용자는 계좌를 조회할수 있을 뿐 다른 계좌로 이체 처리를 할수 없어야 한다.

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

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

●DDL : Data Definition Language 데이터 정의어

👉 데이터베이스 생성과 삭제, 테이블의 생성과 삭제 등 데이터베이스 기본 생성에 관련된 것들로 주로 원타임 실행이 이루어진다.

1. CREATE TABLE - 새로운 테이블 생성

```
CREATE TABLE 테이블 이름(
열이름1 데이터 타입,
```

```
열이름2 데이터 타입,
...
);
```

Quiz4

sample_product 라는 이름의 테이블을 id(숫자) name(글자20), date(날짜타입)

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

```
Table created.
```

```
--날짜 데이터 삽입! 캐스팅 필요!
INSERT INTO sample_product(pid, pname, pdate)
VALUES (1, 'Television', to_date('220810','YYMMDD'));
commit;
```

```
1 row(s) inserted.

Statement processed.
```

2. ALTER TABLE - 테이블 수정

```
ALTER TABLE 테이블이름
  MODIFY(열이름1 데이터타입,
        열이름2 데이터타입,
        ...
        );
```

Quiz5

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

```
ALTER TABLE sample_product
    MODIFY(pname CHAR(15));
```

```
Table altered.
```

⇒ 열이름 수정

```
ALTER TABLE 테이블이름
  RENAME COLUMN 열이름1 TO 열이름2;
```

Quiz6

sample_product 테이블에 있는 PNAME열의 이름을 productname으로 변경하세요.

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

```
Table altered.
```

⇒ 열 삭제

```
ALTER TABLE 테이블이름 DROP COLUMN 열이름;
```

Quiz7

sample_product 테이블에 있는 productname으로 삭제하세요.

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

```
Table altered.
```

3. TRUNCATE TABLE 데이터 삭제

👉 테이블에서 데이터만 전체 삭제 - 테이블에서 구조는 남기고 데이터만 모두 삭제

```
TRUNCATE TABLE 테이블이름;
```

4. DROP TABLE 테이블 완전히 삭제 - 데이터와 구조를 모두 완전히 삭제

👉 DROP TABLE 명령어는 테이블을 삭제할때 모든 자료와 구조까지 삭제하고 사용하던 메모리 까지 돌려준다. DDL 명령어이기 때문에 자동 커밋된다.

```
DROP TABLE 테이블이름;
```

▼ 삭제 명령어 비교

명령어	종류	내용
DELETE	DML	데이터만 삭제
TRUNCATE	DDL	구조는 남겨두고 모두 삭제
DROP	DDL	구조 포함 전체 삭제

VIEW 가상테이블

👉 테이블과 아주 유사하지만 실제 데이터는 없는 가상의 테이블이다.
뷰를 사용하는 이유는 사용자의 편의와 보안문제 때문이다.

> 예를 들어 백화점에서 VIP회원들만 따로 실제 테이블에서 분리하여 가상의 뷰를 만들고
> 담당자를 지정하여 접근할 수 있게 한다면 그 직원은 전체 회원의 정보는 볼 수 없지만,
> 뷰를 통해 VIP회원들의 정보에만 접근하여 다양한 서비스와 프로모션등을 제공할 수 있도록 할 수 있다.

임시로 복잡한 SQL문을 매번 작성하지 않아도 뷰를 한 번 만들어 놓고 사용자를 지정하면 쉽고 편리하게 사용할 수 있다.

- 뷰는 마치 실제 테이블처럼 사용 가능하다.
- 뷰는 자주 쓰는 SQL구문의 결과를 미리 만들어 놓는 개념이다.

- 여러개의 테이블을 조인하여 하나의 뷰를 만들 수 있다.
- 사용자별로 접근 권한을 다르게 설정하여 보안 문제를 해결할 수 있다.

BOOTSTRAP