

데이터베이스 실습 7주차 강의

데이터 검색 : SELECT 문

부속 질의문을 이용한 검색

SELECT 문 안에 또 다른 SELECT 문을 포함하는 질의

표 7-7 다중 행 부속 질의문에 사용 가능한 연산자

연산자	설명
IN	부속 질의문의 결과 값 중 일치하는 것이 있으면 검색 조건이 참
NOT IN	부속 질의문의 결과 값 중 일치하는 것이 없으면 검색 조건이 참
EXISTS	부속 질의문의 결과 값이 하나라도 존재하면 검색 조건이 참
NOT EXISTS	부속 질의문의 결과 값이 하나도 존재하지 않으면 검색 조건이 참
ALL	부속 질의문의 결과 값 모두와 비교한 결과가 참이면 검색 조건을 만족(비교 연산자와 함께 사용)
ANY 또는 SOME	부속 질의문의 결과 값 중 하나라도 비교한 결과가 참이면 검색 조건을 만족(비교 연산자와 함께 사용)

예제 7-43

판매 데이터베이스에서 banana 고객이 주문하지 않은 제품의 제품명과 제조업체를 검색해 보자.

```
▶▶ SELECT 제품명, 제조업체
      FROM 제품
 WHERE 제품번호 NOT IN (SELECT 주문제품
                           FROM 주문
                           WHERE 주문고객 = 'banana');
```

결과 테이블

	제품명	제조업체
1	쿵떡파이	한빛제과
2	매운쫄면	민국푸드
3	얼큰라면	대한식품
4	달콤비스킷	한빛제과

EXISTS : JOIN이 된다.

데이터 삽입 : INSERT 문

데이터 직접 삽입

```
INSERT  
INTO 테이블_이름[(속성_리스트)]  
VALUES (속성값_리스트);
```

예제 7-48

판매 데이터베이스의 고객 테이블에 고객아이디가 tomato, 고객이름이 정은심, 나이가 36 세, 등급이 gold, 적립금은 4,000원, 직업은 아직 모르는 새로운 고객의 정보를 삽입해보자. 그런 다음 고객 테이블에 있는 모든 내용을 검색하여, 삽입된 정은심 고객의 직업 속성이 널 값인지 확인해보자.

▶▶ INSERT

```
INTO 고객(고객아이디, 고객이름, 나이, 등급, 적립금)  
VALUES ('tomato', '정은심', 36, 'gold', 4000);
```

```
SELECT * FROM 고객;
```

결과 테이블

	고객아이디	고객이름	나이	등급	직업	적립금
1	apple	정소화	20	gold	학생	1000
2	banana	김선우	25	vip	간호사	2500
3	carrot	고명석	28	gold	교사	4500
4	orange	김용욱	22	silver	학생	0
5	melon	성원용	35	gold	회사원	5000
6	peach	오형준	(null)	silver	의사	300
7	pear	채광주	31	silver	회사원	500
8	strawberry	최유경	30	vip	공무원	100
9	tomato	정은심	36	gold	(null)	4000

desc 고객; -> describe

부속 질의문을 이용한 데이터 삽입

```
INSERT  
INTO 테이블_이름[(속성_리스트)]  
SELECT 문;
```

데이터 수정 : UPDATE 문

테이블에 저장된 투플에서 특정 속성의 값을 수정

```
UPDATE 테이블_이름  
SET 속성_이름1 = 값1, 속성_이름2 = 값2, ...  
[WHERE 조건];
```

WHERE은 반드시 있어야 함!

(WHERE 절을 생략하면 테이블에 존재하는 모든 투플을 대상으로 수정)

데이터 삭제 : DELETE 문

테이블에 저장된 데이터를 삭제 -> 테이블 삭제 X

```
DELETE  
FROM 테이블_이름  
[WHERE 조건];
```

(WHERE 절을 생략하면 테이블에 존재하는 모든 투플을 삭제해 빈 테이블이 됨)

예제 7-52

주문 테이블에서 주문일자가 2022년 5월 22일인 주문 내역을 삭제해보자. 그런 다음 주문 테이블의 모든 내용을 검색하여 삭제 여부를 확인해보자.

```
▶▶ DELETE  
FROM 주문  
WHERE 주문일자 = '2022-05-22';
```

```
SELECT * FROM 주문;
```

결과 테이블	주문번호	주문고객	주문제품	수량	배송지	주문일자
1 o01	apple	p03	5	서울시 마포구	22/01/01	
2 o02	melon	p01	5	인천시 계양구	22/01/10	
3 o03	banana	p06	45	경기도 부천시	22/01/11	
4 o04	carrot	p02	8	부산시 금정구	22/02/01	
5 o05	melon	p06	36	경기도 용인시	22/02/20	
6 o06	banana	p01	19	충청북도 보은군	22/03/02	
7 o07	apple	p03	5	서울시 영등포구	22/03/15	
8 o08	pear	p02	50	강원도 춘천시	22/04/10	
9 o09	banana	p04	15	전라남도 목포시	22/04/11	

뷰(View)

다른 테이블을 기반으로 만들어진 가상 테이블 (데이터를 실제로 저장하지 않고 논리적으로만 존재하는 테이블)

일반 테이블과 동일한 방법으로 사용

뷰 생성 : CREATE VIEW 문

```
CREATE VIEW 뷰_이름[(속성_리스트)]
AS SELECT 문
[WITH CHECK OPTION];
```

CREATE VIEW 키워드와 함께 생성할 뷰의 이름과 속성의 이름을 나열

AS 키워드와 함께 기본 테이블에 대한 SELECT 문 제시

WITH CHECK OPTION : 제약조건을 지정 (반드시 있어야 함)

예제 7-55

```
고객 테이블에서 등급이 vip인 고객의 고객아이디, 고객이름, 나이, 등급으로 구성된 뷰를  
우수고객이라는 이름으로 생성해보자. 그런 다음 우수고객 뷰의 모든 내용을 검색해보자.
```

```
▶▶ CREATE VIEW 우수고객(고객아이디, 고객이름, 나이, 등급)
      AS SELECT 고객아이디, 고객이름, 나이, 등급
          FROM 고객
         WHERE 등급 = 'vip'
      WITH CHECK OPTION;

      SELECT * FROM 우수고객;
```

뷰 활용

SELECT 문 : 일반 테이블과 같은 방법으로 원하는 데이터를 검색할 수 있음

INSERT, UPDATE, DELETE 문 : 뷰에 대한 삽입·수정·삭제 연산 가능

뷰 삭제 : DROP VIEW 문

```
DROP VIEW 뷰_이름;
```

삽입 SQL의 개념과 특징

삽입 SQL : 프로그래밍 언어로 작성된 응용 프로그램 안에 삽입하여 사용하는 SQL 문