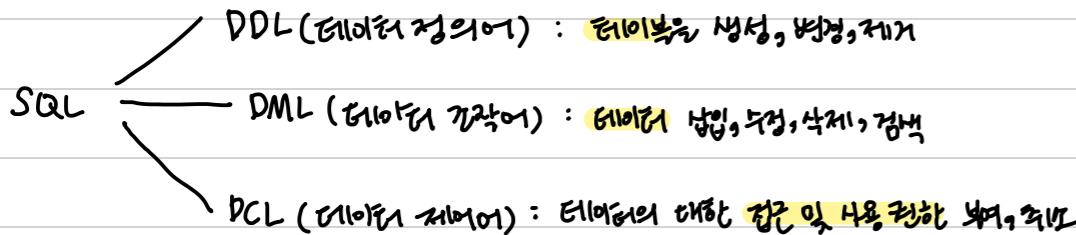


07. 데이터베이스 언어 SQL

01. SQL 소개

SQL은 관계 데이터베이스를 위한 표준 정의어로 비절차적 언어



02. SQL을 이용한 테이블 정의 → 테이블 만들기!

- 테이블 생성

CREATE TABLE 테이블_이름

- ① **필드_이름** **데이터_타입** [NOT NULL] [DEFAULT 기본값]
- ② PRIMARY KEY (필드_이름)
- ③ UNIQUE (필드_이름)
- ④ [FOREIGN KEY (필드_이름) REFERENCES 테이블_이름 (필드_이름)]
[ON DELETE 옵션] [ON UPDATE 옵션]
- ⑤ [CONSTRAINTS 이름] [CHECK(제약)]

① → 세미콜론으로 문장 끝표시

① 필드_이름 데이터_타입 [NOT NULL] [DEFAULT 기본값]

- 테이블을 구성하는 필드들의 이름, 데이터 타입, 넓은 허용 범위, 기본값 필요 예외 정의
↑ 정의
- 테이블 타입에는 INT(또는 INTEGER), SMALLINT, CHAR(n)(또는 CHARACTER(n)), VARCHAR(n), FLOAT(n), TIME, DATE
- [NOT NULL] 설정해주지 않으면 기본값을 허용한다는 의미 (기본값 설정을 반드시 NOT NULL 설정해야 한다 → 가제 무결성 제약규칙)
- 기본값을 명확히 지정해두고 싶다면 DEFAULT 키워드 사용

↑ 기본값 설정

ex) 아이디 VARCHAR(13) NOT NULL ⇒ 아이디란 필드는 최대길이가 13인 가변길이의 문자열로 NULL 값을 허용하지 X

② PRIMARY KEY (주성 키스트)

- 기본키를 지정하는 키워드 → 널값을 가지면 안되기 때문에 NOT NULL
- 기본키는 한 테이블에 1개만 지정 가능

ex) PRIMARY KEY (고객 아이디)

③ UNIQUE (독립 키스트)

- 대체키를 지정하는 키워드 → 유일성, 중복 안족, NULL값 허용
- 대체키는 한 테이블에 여러개 지정 가능

④ [FOREIGN KEY (외성 키스트) REFERENCES 테이블 이름 (속성 간트)]

[ON DELETE 옵션] [ON UPDATE 옵션]

- 외래키를 지정하는 키워드 → 외래키는 다른 테이블의 기본키 or 대체키가 될 수 있다.
- 외래키가 어떤 테이블의 무슨 속성을 참조하는지 REFERENCES 뒤에 적어 → 참고 유일성 제약조건을 지키기 위함!
- ON DELETE / UPDATE NO ACTION : 흑자를 삭제/변경하지 못하게 한다
- ON DELETE / UPDATE CASCADE : 관련 흑자를 함께 삭제/변경 한다
- ON DELETE / UPDATE SET NULL : 관련 흑자의 외래키 값을 NULL로 변경한다
- ON DELETE / UPDATE SET DEFAULT : 관련 흑자의 외래키 값을 초기 설정한 값으로 변경한다

⑤ [CONSTRAINTS 이름] [CHECK (제한)]

- CHECK 키워드를 통해 특정 속성에 제약조건 설정 → CONSTRAINTS로 이름 부여 → 무결성을 위한 표현방식!

ex) 제품 테이블은 제품번호는 제종별로 제종명, 제고량, 단가로 구성, 기본키는 제품번호, 제고량은 양수 0 또는 1000N 이하 유지하는 테이블 생성

CREATE 제품 (

제품번호 CHAR(3) NOT NULL,
제품명 VARCHAR(20),
제고량 INT,
PRIMARY KEY 제품번호,
CHECK (제고량 >= 0 AND 제고량 <= 1000)) ;

• 테이블 변경

① 속성 추가 `ALTER TABLE 테이블_이름`

`ADD 속성_이름 데이터_타입 [NOT NULL] [DEFAULT 기본값];`

② 속성 삭제 `ALTER TABLE 테이블_이름 DROP COLUMN 속성_이름 [CASCADE CONSTRAINTS];`

↳ 삭제할 속성과 관련된 제약조건이나 참조하는 다른 테이블 함께 삭제
작지 않으면, 기본값으로 엔서가져

③ 제약조건 추가 `ALTER TABLE 테이블_이름 ADD CONSTRAINT 제약조건_이름 내용;`

↳ CHECK 명령어 사용 ex) `CHECK (나이 >= 20)`

④ 제약조건 삭제 `ALTER TABLE 테이블_이름 DROP CONSTRAINT 제약조건_이름;`

• 테이블 삭제

`DROP TABLE 테이블_이름 [CASCADE CONSTRAINTS];`

↳ 지정한 테이블을 참조하는 테이블도 함께 지움
작지 않으면, 기본값으로 엔서가져

03. SQL을 이용한 데이터의 조작

• 테이블 조작

↳ 결과 테이블에 중복 허용 (디폴트)
↳ 중복X

`SELECT [All | DISTINCT] 속성리스트`

`FROM 테이블네임;`

- 테이블의 모든 속성을 선택하고 싶을 때는 * 사용
- 결과 테이블에는 다른 속성명을 쓰고 싶을 경우에는 OLD_속성명 AS NEW_속성명
- 속성명 + (+, -, !, *) 등의 부호들을 적어해 결과 테이블이야! 반영 가능

① WHERE 조건

`SELECT [All | DISTINCT] 속성리스트`

`FROM 테이블네임`

`[WHERE 조건];`

- WHERE 조건은 향상된 비교연산자 (>, <, =, <>, >=, <=), 논리연산자 (AND, OR, NOT) 이용해 정석 표현제시

ex) 츄온 테이블에서 apple 고객이 15개 이상 주문한 주문제종, 수량 + 주문일자 정색

`SELECT 주문제종, 수량, 주문일자`

`⇒ FROM 츄온`

`WHERE 주문고객='apple' AND 수량>=15;`

② LIKE를 이용한 검색

문자열을 이용하는 WHERE子의 조건 부분에 사용, 정색 기준을 보통적으로 알고 있을 때 사용

- [] 1. 0개 이상의 문자
- 2. 1개의 문자

Ex) WHERE 고객이름 LIKE '김 . . .' = 김이김씨인 사람

WHERE 고객이아이디 LIKE '_ _ _ _' = 아이디가 5자인 경우

③ NULL을 이용한 검색

[] IS NULL : 특정 속성의 값이 널인지 비교 cf. 나이=NULL X

[] IS NOT NULL : 특정 속성의 값이 널이 아님지 비교

④ 정렬검색

SELECT [ALL | DISTINCT] 뜻대로
FROM 데이터베이스

Ex) 나이를 기준으로 내림차순 정렬

:

[WHERE 조건]
[ORDER BY 뜻대로 [ASC | DESC];

ORDER BY 나이 DESC;

⑤ 집계함수를 이용한 검색

개수, 합계, 평균, 최대값, 최소값 계산 가능 제공

COUNT : 개수 반환
MAX : 최대값
MIN : 최소값
SUM : 합 반환
AVG : 평균 반환

]
숫자 문자

]
숫자데이터만 적용

- NULL 값을 처리하고 계산
- WHERE 조건에는 사용 X, SELECT or HAVING 조건에만 사용

* NULL값은 처리하고 계산하기 때문에 COUNT 함수 적용시 기본기 확장이나 * 이용, DISTINCT 사용하여 중복 제거 가능

⑥ 그룹별 정렬

SELECT [ALL | DISTINCT] 속성리스트

FROM 테이블이름

[WHERE 조건] → 그룹을 나누는 조건

→ 그룹에 대한 조건

[GROUP BY 속성리스트 [HAVING 조건]];

[ORDER BY 속성리스트 [ASC | DESC]];

☞ GROUP BY 속성리스트에 기재되는 속성은 SELECT 문에도 적용되어야 한다!!

• HAVING 절에서의 조건식과 사용 가능

ex) 제품 테이블에서 판매량이 50000 이상인 제품의 개수와 제품명 가장 비싼 대가 정렬

그룹을 나누는 조건
→ 데이터에 대한 조건

SELECT 품목코드, COUNT(*), MAX(단가)
FROM 제품
GROUP BY 품목코드

⑦ 여러 테이블에 대한 조인 정렬

→ 일반적으로 오름차순 사용

여러 테이블을 연결하여 데이터를 정렬하는 것이 조인정렬, 아래 조인정렬 필요! → 속성명은 딱각각이 아니면 같은 이름으로 정렬해야 함

ex) 주문 D.B에서 banana 고기의 조합한 제품의 이름 정렬

⇒ SELECT 제품.제품명
FROM 제품, 주문 → 조인에 활용한 모든 테이블 나열
WHERE 주문.제품코드 = 'banana' ∧ 제품.제품명 = 주문.제품제품명
조인정렬

⑧ 부록 질의문을 이용한 정렬

SELECT 문 안에 다른 SELECT 문이 들어오는 것

→ 조인정렬

→ ① 부록 질의문 (WHERE 문에 정렬하고나, ORDER BY 사용)
②

• 테이블 삽입

INSERT

INTO 테이블_이름 [(속성리스트)]

VALUES (속성값_리스트); : | 관계

• 테이블 수정

UPDATE 테이블_이름

SET 속성_이름1 = 값1, ...

[WHERE 조건]; → 조건에 맞는 희생만 값 수정

적지 않으면 모든 희생이 수정 대상

• 테이블 삭제

DELETE

FROM 테이블_이름

[WHERE 조건];

DROP TABLE

"

→ DELETE 문은 희생을 낸다! 테이블을 낸다! 갖다 다른!

04. 뷰

- 부관 다른 테이블을 기반으로 만들어진 가상 테이블, 실제로 데이터를 저장하지 않음
- 뷰가적으로 만든 존재!
- 뷰를 통해 기본 테이블의 내용을 변경하는 것은 가능하지 않음
- 뷰를 기본으로 새로운 뷰 만들기 가능

• 뷰 정의

CREATE VIEW 뷰_이름 [특성리스트]

AS SELECT 문 → ORDER BY 사용, 나머지는 같음

[WITH CHECK OPTION];

↳ 뷰의 삽입, 수정 연산은 SELECT 문에서 제거한 조건을 위반하면 연산 거부

Ex) 고객 테이블에서 등급이 VIP인 고객의 아이디, 이름, 나이로 구성된 뷰를 유도고객이란 이름으로 생성

⇒ CREATE VIEW 유도고객(고객아이디, 이름, 나이)
AS SELECT 고객아이디, 이름, 나이
FROM 고객
WHERE 등급 = 'VIP'
WITH CHECK OPTION;

] 기본 SELECT 문

• 뷰의 활용

뷰에서 SELECT, INSERT, UPDATE, DELETE 문 실행 가능 → 아래 예외사항이 아니라 연산이 수행되어 기본 테이블도 변경된 경우는 뷰에 없는 쪽에는 NULL 저장

↳ 기본 테이블도 변한다. 단! 모든 연산이 허용되는 것은 X

- ① 뷰가 기본 테이블의 기본기를 포함하고 있지 않은 경우
- ② 뷔가 커테이ning에 의해 계산된 값을 포함하고 있는 경우
- ③ DISTINCT(중복제거X) 기워드를 포함한 뷔
- ④ GROUP BY 절을 포함하여 정의한 뷔인 경우
- ⑤ 여러 테이블과 조인하여 정의한 뷔인 경우

- 뷰의 장점

- ① 질의문을 풍미 쉽게 작성 가능
- ② 데이터의 보안 유지
- ③ 데이터 관리의 편함

- 뷰의 단점

- `DROP VIEW 뷰_이름;`

05. 삽입 SQL

- 유통 프로그램 (C, C++, JAVA) 안에 삽입하여 사용하는 SQL로 → 일에 EXEC SQL 작성

- 작업하는 도구 사용

→ 수행 결과로 반환된 행들을 향상에 대해서 기록할 때와, 예외 행을 결과로 반환하는 SELECT문에서 사용