■ 기본 SQL 작성하기

▪ 가. DDL(데이터 정의어)

DDL(Data Definition Language)은 데이터를 정의하는 언어

1) DDL 관련 용어

가) 스키마(Schema) : DBMS 특성과 구현 환경을 감안한 데이터 구조

나) 도메인(Domain) : -속성이 가질 수 있는 값의 범위

다) 테이블(Table) : 데이터 저장 공간

라) 뷰(View) : 하나 이상의 물리 테이블에서 유도되는 가상의 논리 테이블

마) 인덱스(Index) : 검색을 빠르게 하기 위한 데이터 구조

2) DDL 명령어

가) CREAT : 데이터베이스 오브젝트 생성

나) ALTER : 데이터베이스 오브젝트 변경

다) DROP : 데이터베이스 오브젝트 삭제

3) DDL 활용 예제

```
CREATE TABLE 고객 (
    고객ID int NOT NULL,
    고객명 varchar(10),
    적립금 varchar(10),
    PRIMARY KEY (고객ID)
);
CREATE TABLE 구매 (
    고객ID int NOT NULL,
    구매일자 varchar(10),
    결재방식 varchar(10),
    ISBN varchar(10),
    PRIMARY KEY (고객ID),
    FOREIGN KEY (고객ID) REFERENCES 고객(고객ID) ON DELETE CASCADE
);
```

※ 모든 테이블 검색 : SELECT * FROM tab;

※ 테이블 필드/속성 확인(오라클) : DESC 테이블명;

※ 자동 증가 : AUTO_INCREMENT

※ 테이블 생성 확인 : DESC 고객

※ 제약조건 (CONSTRAINT) : 무결성을 지키기 위해 제한된 조건을 의미함.

ex) CONSTRAINT 고객_구매_fk FOREIGN KEY (고객ID) REFERENCES 고객(고객ID)

- NOT NULL : NULL값을 허용하지 않는다.
- UNIQUE : 중복된 값을 가질수 없는 유일한 값이다.
- CHECK : 조건에 맞는 data만 허용한다.
- PRIMARY KEY : 반드시 존재해야하며, 유일한 값이어야 한다.
- FOREIGN KEY : 다른 테이블에서 키값을 참조한다.
- DEFAULT : 기본값을 가진다.

```
ALTER TABLE 고객 ADD 메일 varchar(255);
ALTER TABLE 고객 DROP COLUMN 메일;
DROP TABLE 고객;
```

▪ 나. DML (데이터 조작어)

데이터를 조작하는 명령어를 DML(Data Manipulation Language)이라고 한다.

1) DML 명령어

가) INSERT : 테이블의 내용을 삽입

나) SELECT : 테이블의 내용을 조회

다) UPDATE : 테이블의 내용을 변경

라) DELETE : 테이블의 내용을 삭제

2) DML 활용 예제

```
INSERT INTO 고객 (고객ID, 고객명, 적립금) VALUES ('1', '홍길동', '100')
INSERT INTO 고객 VALUES (custno_seq.nextval, '홍길동', '200');  ※ 시퀀스(custno_seq)가 설정된 경우
SELECT * FROM 고객;    ※ 모든 테이블 검색 : SELECT * FROM tab;
SELECT 고객명, 적립금 FROM 고객 WHERE 고객ID=1;
UPDATE 고객 SET 고객명= '장발장', 적립금= '1000' WHERE 고객ID = 1;
DELETE FROM 고객 WHERE 고객ID=1;
```

▪ 다. DCL (데이터 제어어)

데이터베이스에서 데이터 이외의 오브젝트에 대해 조작할 필요가 있다. 이때 사용하는 SQL 명령을 DCL(Data Control Language)이라고 한다.

1) DCL 제어 대상

가) 사용자 권한 : 사용자를 등록하고, 사용자에게 특정 데이터베이스를 사용할 수 있는 권리를 부여하는 작업

나) 트랜잭션 : 안전한 거래 보장으로 동시에 다수의 작업을 독립적으로 안전하게 처리하기 위한 상호 작용 단위

2) DCL 명령어

가) GRANT : 사용자 권한 부여

나) REVOKE : 사용자 권한 회수

다) COMMIT : 트랜잭션 확정

라) ROLLBACK : 트랜잭션 취소

3) DCL 활용 예제

※ 테스트를 위해 새로운 계정(아이디)을 만듬 : test

```
- CONNECT system/1234;
- CREATE USER test IDENTIFIED BY 1234;
- GRANT CREATE SESSION TO test;
- GRANT CREATE TABLE TO test;
- GRANT SELECT ANY TABLE TO test;
```

※ 모든 권한 부여 : GRANT CONNECT, RESOURCE, DBA TO test;

```
- REVOKE CREATE SESSION FROM test;
- INSERT INTO 고객 VALUE ('2', '홍길동', '100');
COMMIT;
- SELECT * FROM 고객;
- DELETE FROM 고객 WHERE 고객명='홍길동' ;
ROLLBACK;
- SELECT * FROM 고객;
```

▪ 라. 데이터 사전

데이터 사전(Data Dictionary)에는 데이터베이스의 데이터를 제외한 모든 정보가 있다. 데이터 사전의 내용을 변경하는 권한은 시스템이 가지며, 사용자에게는 읽기 전용 테이블 형태로 제공되므로 단순 조회만 가능하다.

1) 데이터 사전 내용
   - 가) 사용자 정보(아이디, 패스워드 및 권한 등)
   - 나) 데이터베이스 객체 정보(테이블, 뷰, 인덱스 등)
   - 다) 무결성 제약 정보
   - 라) 함수, 프로시저 및 트리거 등

2) 데이터 사전 영역
   - 가) DBA : 데이터베이스의 모든 객체 조회 가능 (DBA_는 시스템 접근 권한 의미)
   - 나) ALL : 자신의 계정으로 접근 가능한 객체와 타 계정의 접근 권한을 가진 모든 객체 조회 가능
   - 다) USER : 현재 자신의 계정이 소유한 객체 조회 가능

3) 데이터베이스 관리 데이터 종류
   - 가) 저장 데이터 : 컴퓨터를 통해 접근 가능한 저장 매체에 저장된 데이터
   - 나) 통합 데이터 : 중복이 최소화된 데이터
   - 다) 공유 데이터 : 여러 응용 프로그램이 공동으로 사용하는 데이터
   - 라) 운영 데이터 : 조직의 목적을 위해 존재 가치가 확실하고 반드시 필요한 데이터

■ 고급 SQL 작성하기

▪ 가. 인덱스

인덱스는 데이터를 빠르게 찾을 수 있는 수단으로서, 테이블에 대한 조회 속도를 높여 주는 자료구조를 일컫는다. 세부적으로 정규화되지 않은 테이블에서 필요한 속성(컬럼)부분만 먼저 추출 하고싶을 때 사용한다. SELECT문과 비슷하지만 DDL 명령어를 사용하여 테이블을 실제로 만들어, 삭제한다.

1) 인덱스 생성

CREATE INDEX <index_name> ON <table_name> (<column(s)>);

2) 인덱스 삭제

DROP INDEX <index name>;

3) 인덱스 변경

ALTER INDEX <index name> ON <table name> (<column(s)>);

▪ 나. 뷰(View)

뷰는 사용자의 관점에서 필요한 부분만 골라서 논리적으로 만든 가상 테이블을 의미한다.

1) 뷰의 장점

가) 논리적 독립성 제공

논리(가상) 테이블로 테이블의 구조가 변경되어도 뷰를 사용하는 응용 프로그램은 변경하지 않아도 됨.

나) 사용자 데이터 관리 용이

복수 테이블에 존재하는 여러 종류의 데이터에 대해 단순한 질의어 사용이 가능

다) 데이터 보안 용이

중요 보안 데이터를 저장 중인 테이블에는 접근 불허하고, 해당 테이블의 일부 정보만을 볼 수 있는 뷰에는 접근을 허용하는 방식으로 보안 데이터에 대한 접근 제어 가능

2) 뷰의 단점

가) 뷰 자체 인덱스 불가

인덱스는 물리적으로 저장된 데이터를 대상으로 하기에 논리적 구성인 뷰 자체는 인덱스를 가지지 못함.

나) 뷰 정의 변경 불가

뷰의 정의를 변경하려면 뷰를 삭제하고 재생성하여야 함.

다) 데이터 변경 제약 존재

뷰의 내용에 대한 삽입, 삭제, 변경 제약이 있음.

3) 뷰 사용 예시

CREATE VIEW <뷰이름> AS select * from <테이블명>;

SELECT * FROM <뷰이름>;

DROP VIEW <뷰이름>;

▪ 다. 다중 테이블 검색

데이터를 분해하는 방법으로 정규화 기법이 사용되며, 통합하는 기법으로 다중 테이블에 대한 검색이 사용된다.

1) 다중 테이블 사용 기법

가) 조인 : 두 개의 테이블을 결합하여 데이터를 추출하는 기법

나) 서브쿼리 : SQL문 안에 포함된 SQL문 형태의 사용 기법

다) 집합연산 : 테이블을 집합 개념으로 조작하는 기법

2) 조인

가) 내부조인(INNER JOIN)

- 정의 : 두 테이블에 공통으로 존재하는 필드값을 이용하여 테이블을 합하는 방법

(1) 내부조인(INNER JOIN) : 조건(WHRER)과 일치하는 데이터를 중심으로 테이블을 합한다.

[명시적 표현법]

```
SELECT *
FROM 고객
INNER JOIN 구매
ON 고객.고객아이디 = 구매.고객아이디;
```

[암묵적 표현법] --->> 일반적으로 가장 많이 사용하는 방법

```
SELECT *
FROM 고객, 구매
WHERE 고객.고객아이디 = 구매.고객아이디;
```

(2) 자연조인(NATURAL JOIN) : 내부조인에서 중복된 속성을 하나로 출력 (중복제거)

[NATURAL JOIN]

```
SELECT * FROM 고객 NATURAL JOIN 구매;
```

나) 외부조인(OUTER JOIN)

- 정의 : 공통의 존재하는 필드값이 없어도 null 값으로 두테이블을 하나로 합하는 방법

(1) 왼쪽 외부조인(LEFT OUTER JOIN)

내부조인에서 왼쪽 테이블의 기준으로 공통의 데이터가 없더라도 null값으로 테이블을 생성한다.

[LEFT OUTER JOIN]

```
SELECT * FROM 고객 LEFT OUTER JOIN 구매 ON 고객.고객아이디 = 구매.고객아이디;
```

(2) 오른쪽 외부조인(RIGHT OUTER JOIN)

내부조인에서 오른쪽 테이블의 기준으로 공통의 데이터가 없더라도 null값으로 테이블을 생성한다.

[RIGHT OUTER JOIN]

```
SELECT * FROM 고객 RIGHT OUTER JOIN 구매 ON 고객.고객아이디 = 구매.고객아이디;
```

3) 서브쿼리

가) GROUP BY 를 이용한 그룹화

- 정의 : 테이블의 특정 속성으로 그룹화하여 집계함수(sum, avg, count 등)를 이용하여 테이터를 계산함.

[적립금에 따른 고객수]

```
SELECT 적립금, COUNT(*) AS 고객수 FROM 고객 GROUP BY 적립금;
```

[적립금에 따른 고객수가 2명이상인 고객수

```
SELECT 적립금, COUNT(*) AS 고객수
FROM 고객
WHERE 고객ID > 10          // WHERE 절은 특별한 조건이 없을 경우 생략가능
GROUP BY 적립금
HAVING COUNT(*) >= 2;     // 반드시 GROUP BY와 함께 사용되며, 그룹의 조건을 명시
```

나) SQL 다중처리문(서브쿼리)을 이용한 그룹화

- 정의 :테이블의 특정 속성으로 그룹화하여 집계함수(sum, avg, count 등)를 이용하여 테이터를 계산함.

['20200405'일자에 구매한 고객명 검색]

```
SELECT 고객명
FROM 고객
WHERE 고객ID = ( SELECT 고객ID FROM 구매 WHRER 구매일자 = '20200405');
```

[구매활동을 하지 않은 고객정보 검색]

```
SELECT *
FROM 고객
WHERE 고객ID NOT IN ( SELECT 고객ID FROM 구매 );
```

※ ORDER BY 를 이용한 정렬(내림차순, 오름차순)

- 정의 : 테이블의 값을 내림차순(4, 3, 2, 1) 또는 오름차순(1, 2, 3, 4, 5)으로 정렬하여 검색

[고객 테이블에서 고객명(오름차순), 적립금(내림차순)으로 검색]

```
SELECT 고객명, 적립금 FROM 고객 ORDER BY 고객명 ASC, 적립금 DESC;
```

4) 집합 연산자의 종류

-UNION

(합집합)여러 개의 SQL문의 결과에 대한 합집합으로 결과에서 모든 중복제거

-UNION ALL

(합집합)여러 개의 SQL문의 결과에 대한 합집합으로 중복된 행도 그대로 결과로 표시된다. 즉, 단순히 결과만 합쳐놓은 것이다. 일반적으로 여러 질의 결과가 상호 배타적인(Exclusive)일때 많이 사용한다. 개별 SQL문의 결과가 서로 중복되지 않는 경우, UNION과 결과가 동일하다. (결과의 정렬 순서에는 차이가 있을 수 있음)

-INTERSECT

(교집합)여러 개의 SQL문의 결과에 대한 교집합이다. 중복제거

-EXCEPT

(차집합)앞의 SQL문의 결과에서 뒤의 SQL문의 결과에 대한 차집합이다. 중복제거 (일부 데이터베이스는 MINUS를 사용함)