1. 데이타베이스 검색

☞ **DML(Data Manipulation Language)**

• SELECT문 : 테이블로부터 행을 검색한다.

• INSERT문 : 테이블에 행을 추가한다.

• UPDATE문 : 테이블에 칼럼 값을 수정한다.

• DELETE문 : 테이블에 불필요한 행을 삭제한다.

• MERGE문 : 두 개 이상을 테이블 데이터를 하나의 테이블로 병합한다.

2. SELECT문

2.1. SELECT문의 기본 형식

SELECT문은 테이블로부터 필요한 데이터를 질의(query)하여 검색하는 명령문으로, 두 개의 필수절로 구성된다.

```
SELECT [ DISTINCT | UNIQUE ] {칼럼명 [별명], .....}, *
FROM 테이블명;
```

SELECT문의 SELECT절과 FROM절은 필수절이다. SELECT문은 FROM절에 기술된 테이블로부터 SELECT절에 기술된 칼럼 등의 내용을 검색하여 화면에 출력한다.

☞ SELECT절에는

- FROM정의 테이블로부터 검색하여 출력할 칼럼명이나, '*'(모든 칼럼 표시)를 기술한다. 여러 칼럼을 기술할 때는 콤마(,)로 구분한다.
- 칼럼명 앞에 기술하는 DISTINCT 또는 UNIQUE는 중복되지 않은 칼럼값을 출력한다.
- 칼럼명 뒤에는 한 칸 이상의 공백을 주고 별명을 줄 수 있다.
- 산술식, 리터럴, 함수 등을 기술할 수 있다.

☞ FROM절에는 검색에 사용할 객체명, 즉 테이블명이나 뷰명을 기술한다.

☞ 결가 행을 출력할 때 SELECT절의 칼럼명이나 수식 등을 부제목으로 사용한다.

1) 기본 맞춤 형식

출력하는 칼럼 데이터의 기본 맞춤 형식은 문자형과 날짜형 데이터는 왼쪽 정렬되고, 숫자형 데이터는 오른쪽 정렬되어 출력된다.

2) 중복되지 않은 값 검색

DISTINCT 또는 UNIQUE는 칼럼에 대하여 중복되지 않은 값을 출력하는 예약어이다.

예) 1. Department 테이블의 모든 데이터 조회

2. Professor 테이블의 중복되지 않은 Dept_ID 칼럼의 값들을 조회[1명 이상의 교수가 존재하는 학과명 조회]

2.2 SELECT문의 확장

```
SELECT 칼럼명1, 칼럼명2, ....., *, 리터럴, 함수, 수식, .....
FROM 테이블명1, 테이블명2, 뷰명1, .....
WHERE 검색조건1.....
GROUP BY 칼럼명1, 칼럼명2, .....
HAVING 검색조건2
ORDER BY 칼럼명1 [ ASC | DESC ]
                순서번호1 [ASC | DESC ]
```

☞ SELECT문의 주요 사항

- SELECT절과 FROM절은 필수절이고, 나머지 절은 선택절이다.
- 절의 기술 순서는 고정이다.
- HAVING절은 GROUP BY절이 기술될 때만 사용할 수 있다.
- SELECT문의 결과 행의 수는 0행, 한 행, 복수 행 모두 정상적으로 실행된다.
- SELECT문의 실행 결과는 화면에 출력된다.

☞ 별명(alias) 사용

SELECT절과 FROM절에 별명(Alias)을 사용할 수 있다. 별명은 SELECT절에 기술되는 복잡한 칼럼명이나 산술식, 함수 등에 대하여 간단하고, 이해하기 쉽도록 하기 위해서 별명을 사용한다.

• SELECT절의 칼럼 별명

SELECT절의 별명은 한 칸 이상의 공백을 띄운 후 " " 내에 별명을 기술하거나 AS 다음에 기술한다.

√ 칼럼의 별명은 출력시 부제목으로 사용된다.

√ 칼럼의 별명은 ORDER BY절의 출력순서를 지정할 때 사용할 수 있다.

```
SELECT Student_ID "학번", Name as 이름, .....
```

- **테이블명의 별명**

FROM절에 기술하는 테이블의 별명은 단순화하기 위하여 사용한다. FROM절에 테이블명 한 칸 이상의 공백을 띄운 후 별명을 기술한다.

√ 테이블명의 별명은 SELECT문의 각 절에서 칼럼명을 구분할 때 사용한다.

```
FROM Department D, ........
```

예) 1. Professor 테이블의 중복되지 않은 Dept_ID 칼럼의 값들을 조회. 단 부제목은 "소속하과"로 조회.

2. Course 테이블에서 과목코드(Course_ID), 과목(Title), 학점 수(C_Number)를 조회

3. ORDER BY절에 출력순서 지정

ORDER BY절은 선택절이며, 검색된 결과 행의 출력순서를 지정한다. 출력순서를 지정하는 정렬 방법은 오름차순 정렬과 내림차순 정렬의 두 가지가 있다.

☞ 오름차순 정렬(ascending Sort)

출력순서를 지정하는 분류 키(sort key)를 작은 값부터 큰 값으로 정렬하는 것으로, 숫자형은 0부터 9, 영문자는 A부터 Z, 한글은 '가나다.....'순으로 정렬된다.

☞ 내림차순 정렬(descending Sort)

출력순서를 지정하는 분류 키를 큰 값부터 작은 값으로 정렬하는 것으로, 숫자형은 9부터 0, 영문자는 Z부터 A, 한글은 '하파타....'순으로 정렬된다.

☞ ORDER BY절

```
ORDER BY 칼럼명1 [ASC | DESC], 칼럼명2 [ASC | DESC], .............
                순서번호1 [ACS | DESC], 순서번호2 [ACS | DESC], ..................
```

FROM절에 기술한 테이블의 컬럼명이나, SELECT절에 기술된 칼럼명, 산술식 등의 순서번호로 출

력되는 데이터를 정렬할 수 있다. ORDER BY절에는 테이블의 칼럼명, 순서번호, SELECT절의 별명은 기술할 수 있으나, 수식이나 함수, 리터럴 등은 기술할 수 없다.

정렬 방법 지정은 칼럼명이나 순서번호를 우선순위로 기술하고, 한 칸 이상의 공백을 띄운 후, 오름차순 정렬은 ASC, 내림차순 정렬은DESC를 기술한다. 단, 오름차순 정렬 방법은 생략할 수 있다.

☞ **산술연산자**

SELECT문에 수식을 사용할 수 있다. 숫자형 연산에 사용되는 산술연산자는 +(더하기), -(빼기), *(곱하기), /(나누기)가 있다. 특히, SELECT절에 기술된 칼럼명 이외에 수식이나, 함수명 등의 값으로 정렬하고자 할 때, ORDER BY절에 수식, 또는 함수명은 기술할 수 없다.

예) Course 테이블에서 과목코드(Course_ID), 과목명(Title), 학점 수(C_Number), 과목별 수강료(학점수 * 30000 + 추가수강료)를 조회하되, 과목별 수강료를 내림차순, 과목 코드는 오름차순으로 조회한다.

4. WHERE절에 검색조건 지정

일부 텡ㅣ버나 특정 데이터를 검색하기 위한 조건을 지정할 때 사용한다. 테이블의 행들은 WHERE절의 검색조건이 참(true)이 되는 행에 대해서만 반환된다.

```
WHERE   expr1 연산자 expr2
```

• expr1과 expr2는 칼럼명, 리터럴, 수식, 함수 등이 기술될 수 있고— 데이터타입이 반드시 같아야 하며, 두 개의 표현식에 대하여 검색 조건이 참(true)이 되는가를 확인한다.

• 연산자는 산술연산자, 관계 연산자, 논리 연산자, SQL 연산자, IS NULL, 연산자 등이 있다.

4.1. 연산자의 종류

☞ **산술연산자**

산술연산자	의 미	연산 우선 순위	표현식
+	더하기	2	expr1 + expr2
-	빼기	2	expr1 - expr2
*	곱하기	1	expr1 * expr2
/	나누기	1	expr1 / expr2

예) SQL> SELECT 125* 32+12500 FROM DUAL;

☞ 관계연산자

관계연산자	의 미	표현식
>	크다	expr1 > expr2
>=	크거나 같다	expr1 >= expr2
<	작다	expr1 < expr2
<=	작거나 같다	expr1 <= expr2
=	같다	expr1 = expr2
<>, !=	같지 않다	expr1 != expr2

예) 1. Professor 테이블로부터 "컴공" 학과의 교수명을 조회.

2. Course 테이블로부터 추가 수강료가 30000원 이상인 과목명을 출력하되, 추가 수강료를 내림차순으로 조회.

☞ 논리연산자

논리연산자	의 미	표현식
AND	논리곱	조건식1 AND 조건식2
OR	논리합	조건식1 OR 조건식2
NOT	부정	NOT 조건식

예) Student테이블에서 '컴공'학과 2학년 학생의 학과, 학년, 성명을 조회

☞ 연결연산자

연결연산자 '||'(concatenation)는 문자형 데이터들을 결합한다.

```
expr1 || expr2
```

예) Professor테이블로부터 소속학과, 교수명, 전화번호를 자연어로 조회하되, 학과코드(Dept_ID) 순으로 조회.

☞ LIKE 연산자

부분적으로 알고 있는 문자열의 문자패턴에 의해서 검색할 수 있는 SQL 연산자이다. Expr은 칼럼명을 기술하고, '문자패턴'에는 검색할 문자열을 기술한다. Expr이 '문자패턴'과 일치하면 참이 된다. 부정은 NOT LIKE 연산자이다.

```
expr LIKE '문자패턴'
```

검색할 '문자패턴'에 사용되는 대체 문자는 두 가지가 있다.

① % 기호는 모든 문자를 대체할 때 사용하고,

② _(underbar) 기호는 한 문자를 대체할 때 사용한다.

예를 들어, Name 칼럼에서 '홍'씨 성을 모두 검색할 경우에는 WHERE절에 Name LIKE '홍%', '길동'의 이름을 검색할 경우에는 Name LIKE '_길동'으로 대체문자를 사용하여 문자패턴을 기술한다.

예) Student테이블로부터 '이'씨 성의 학생 명단 조회.

☞ IN 연산자

가능한 값들의 목록과 비교하여, 가능한 값의 목록 중 하나일 대 참(true)이 되고, 그렇지 않으면 거짓(false)이 되는 SQL 연산자이다. 부정은 NOT IN 연산자이다.

```
expr IN (값1, 값2, ....)
```

• expr은 칼럼명이나, 수식, 함수 등을 기술하고, 가능한 값들의 목록을 (값1, 값2, 값3.....)에 기술한다.

예) Professor 테이블에서 학과 코드(Dept_ID)가 '컴공', '정통'학과에 재직 중인 교수의 명단을 학과코드순으로 조회.

☞ BETWEEN~AND~ 연산자

지정된 범위의 최소값부터 최대값 이내의 범위에 포함되는가를 검색하는 SQL 연산자이다. 범위 내의 값이면 참이 되고, 그렇지 않으면 거짓이 된다ㅏ. 비교할 값은 숫자형, 문자형, 또는 날짜명 데이터를 모두 가능하다. 부정은 NOT BETWEEN~AND~ 연산자이다.

```
expr BETWEEN 최소값  AND 최대값
```

• expr은 칼럼명이나 수식, 함수 등을 기술하고 최소값과 최대값은 숫자형 데이터나 문자형 데이터, 또는 날짜형 데이터에 대해서 검색할 범위 값을 기술한다.

예) SG_Sources 테이블에서 성적(Score)이 90점부터 94점까지의 범위 내 성적을 성적순으로 조회

☞ IS NULL 연산자

expr 값이 Null이면 참이 되고, 그렇지 않으면 거짓이 된다. 부정은 IS NOT NULL 연산자이다.

```
expr IS NULL
```

예) Course 테이블로부터 추가수강료가 널(Null)인 행을 검색하여 조회.