ORACLE

Introduction to Oracle9i
: SQL

DBGUIDE.net
데이터베이스 가이드 네트워크

ORACLE

Chapter1.
기초 SELECT문 작성

- 가장 기본적인 SELECT문을 실행한다.
- 조건에 맞는 데이터를 검색한다.

1. 기초 SELECT 구문

```
SELECT    * | 칼럼명 | 표현식
FROM      테이블명;
```

(1) 테이블 전체를 검색할 경우

```
SELECT    *
FROM      employees ;
```

(2) 테이블의 특정 컬럼을 검색할 경우

```
SELECT    employee_id, last_name
FROM      employees ;
```

2. SQL문 작성 지침

- SQL문은 대소문자를 구분하지 않음
- SQL문은 한 행 또는 여러 행에 걸쳐 작성 가능
- 예약어는 단축하거나 행을 바꿔서 사용 불가

3. 산술표현식

(1) 연산자 종류

연산자	의미
+	더하기
–	빼기
*	곱하기
/	나누기

※ 연산자 우선순위

1) 곱하기 및 나누기가 더하기나 빼기보다 먼저 수행
2) 동등한 우선순위를 갖는 연산은 왼쪽에서부터 수행
3) 괄호를 사용해서 연산의 우선순위 조정 가능

3. 산술표현식

(2) 연산자 사용의 예

```
SELECT    last_name, 12*salary+100
FROM      employees ;
```

```
SELECT    last_name, 12*(salary+100)
FROM      employees ;
```

4. NULL 값

(1) NULL 값의 정의

- 사용할 수 없고, 지정되지 않고, 알려져 있지 않고, 적용할 수 없는 값
- NULL은 0 또는 space와는 다른 개념

(2) 산술 연산에서의 NULL 값

- 연산하고자 하는 값 중 하나라도 NULL 값을 가지면 연산의 결과는 무조건 NULL

```
SELECT     last_name, 12 * salary  * commission_pct
FROM       employees ;
```

5. 컬럼의 별칭

(1) 별칭의 정의

- 컬럼의 원 이름이 아닌 별도의 이름
- 주로 연산이 수행된 컬럼에 사용
- 컬럼명 다음에 AS를 쓴 뒤 그 뒤에 별칭으로 사용할 단어를 제시. AS는 생략 가능
- 특수문자나 공백을 칼럼의 별칭으로 사용하고자 하면 별칭에 큰따옴표를 사용

(2) 컬럼 별칭 사용의 예

```
SELECT    last_name AS name, employee_id empid, salary*12 "Annual Salary"
FROM      employees ;
```

6. 연결 연산자

(1) 연결연산자

- 컬럼 또는 문자를 다른 컬럼과 연결할 때 사용
- || 기호로 표현

(2) 연결연산자 사용의 예

```
SELECT    last_name || department_id AS “Employees”
FROM      employees ;
```

```
SELECT    last_name || ‘is in’ || department_id AS “Employees”
FROM      employees ;
```

7. 중복 행의 제거(DISTINCT)

(1) DISTINCT

- 같은 컬럼에 있는 동일한 값은 한번만 출력
- SELECT 바로 뒤에 위치

(2) DISTINCT 사용의 예

```
SELECT     DISTINCT   department_id
FROM       employees ;
```

8. 행의 제한(WHERE 절)

(1) WHERE 절

- 검색 결과에 대한 제한을 둘 때 WHERE절에 제시
- FROM 절 바로 다음 절

(2) WHERE 절 사용의 예

```
SELECT    employee_id, last_name
FROM      employees
WHERE     department_id = 90;
```

9. 문자값 및 날짜값

(1) 문자 및 날짜값 비교시 고려사항

- 문자 및 날짜값은 반드시 작은 따옴표 안에 작성하여 비교
- 문자값은 대소문자를 구분
- 날짜값은 형식을 구분
- 날짜값의 기본 형식은 DD-MON-RR

(2) 문자 및 날짜값 비교의 예

```
SELECT     employee_id, job_id
FROM       employees
WHERE      last_name = 'KING' ;
```

10. 비교연산자

(1) 연산자의 종류

연산자	의미
=	같다
>	~보다 크다
>=	~보다 크거나 같다
<	~보다 작다
<=	~보다 작거나 같다
<>	같지 않다

(2) 비교연산자 사용의 예

```
SELECT     employee_id, last_name, salary
FROM       employees
WHERE      salary >= 2500  ;
```

11. 기타 비교연산자

(1) 기타 비교연산자의 종류

연산자	의미
BETWEEN ... AND ...	두 값 사이의 값
IN	여러 값 중 하나와 일치하는 값
LIKE	문자의 패턴이 일치하는 값
IS NULL	값이 NULL인 값

11. 기타 비교연산자

(2) 비교연산자 사용의 예

1) BETWEEN … AND …

```
SELECT     employee_id, last_name, salary
FROM       employees
WHERE      salary BETWEEN 2000 AND 3000  ;
```

2) IN

```
SELECT     employee_id, last_name, salary
FROM       employees
WHERE      salary IN (2000, 2500, 3000)  ;
```

11. 기타 비교연산자

(2) 비교연산자 사용의 예

3) LIKE

- Wildcard 검색 수행
- 검색조건은 다음 문자 중 하나를 포함
  - % : 0 또는 많은 문자
  - _ : 하나의 문자
- % 또는 _ 등의 문자를 비교하고자 할 때는 ESCAPE 사용

```
SELECT     employee_id, last_name
FROM       employees
WHERE      last_name  LIKE  '_a%' ;
```

11. 기타 비교연산자

(2) 비교연산자 사용의 예

4) IS NULL

```
SELECT     employee_id, last_name
FROM       employees
WHERE      manager_id  IS NULL ;
```

12. 논리연산자

(1) 논리연산자의 종류

연산자	의미
AND	양쪽 조건 모두 TURE이면 TRUE를 반환
OR	양쪽 조건 중 하나만 TURE이면 TRUE를 반환
NOT	조건이 FALSE일 경우 TRUE반환

12. 논리연산자

(2) 논리연산자 사용의 예

1) AND

```
SELECT     employee_id, last_name, salary
FROM       employees
WHERE      salary > 8000
AND        last_name LIKE '%K%' ;
```

2) OR

```
SELECT     employee_id, last_name, salary
FROM       employees
WHERE      salary > 8000
OR         last_name LIKE '%K%' ;
```

12. 논리연산자

(2) 논리연산자 사용의 예

3) NOT

```
SELECT     employee_id, last_name, job_id
FROM       employees
WHERE      job_id  NOT  IN  ('MK_REP', 'MK_MAN') ;
```

13. 우선순위의 규칙

실행 순서	연산자
1	산술연산자
2	연결연산자
3	비교연산자
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	NOT
7	AND
8	OR

- **위의 우선순위에 대한 규칙은 괄호를 사용해서 조절 가능**

14. 데이터의 정렬

(1) ORDER BY 절

1) 오름차순 또는 내림차순으로 정렬

- ASC : 오름차순 (A→Z순, ㄱ→ㅎ순, 1→10순)
- DESC : (Z→A순, ㅎ→ㄱ 순, 10→1순)

2) ORDER BY 절은 항상 SELECT문의 마지막 절에 작성

3) SELECT 절에 없는 컬럼으로도 정렬이 가능

```
SELECT     employee_id, last_name, salary
FROM       employees
ORDER BY  salary  ;
```

```
SELECT     employee_id, last_name, salary
FROM       employees
ORDER BY  salary  DESC;
```

14. 데이터의 정렬

(2) 컬럼의 별칭을 통한 정렬

```
SELECT     employee_id, last_name name
FROM       employees
ORDER BY   name  ;
```

(3) 여러 컬럼을 통한 정렬

```
SELECT     employee_id, last_name, department_id, salary
FROM       employees
ORDER BY   department_id, salary DESC, last_name ;
```