2장 데이터 제한과 정렬

목적

DB로 부터 데이터를 검색하는 동안 디스플레이되는 데이터 행을 축소하거나 행의 순서를 명시.

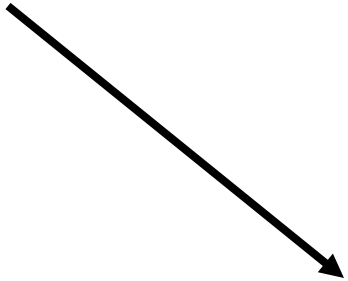
- 질의에 의해 검색되는 행을 제한
- 질의에 의해 검색되는 행을 정렬

Selection을 사용하여 행 제한

EMP

EMPNO	ENAME	JOB	.....	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
7566	JONES	MANAGER		20
.....		....		

"...부서 10의 모든 종업원을 검색"



EMP

EMPNO	ENAME	JOB	.....	DEPTNO
7839	KING	PRESIDENT		10
7782	CLARK	MANAGER		10
7934	MILLER	CLERK		10

선택된 행 제한

- **WHERE**절을 사용하여 리턴 되는 행을 제한 한다.

```
SELECT    [DISTINCT]  {*, column  [alias], …..}
FROM       table
[WHERE    condition(s)];
```

- **WHERE**절은 **FROM**절 다음에 온다.

- WHERE : 조건을 만족하는 행으로 질의를 제한

- condition : 열 이름, 표현식, 상수 그리고 비교 연산자로 구성

- 열, 리터럴 값, 산술 표현식 또는 함수 값을 비교 할수 있다.

WHERE 절 사용

```
SQL> SELECT   ename,  job,  deptno
2       FROM      emp
3       WHERE     job = 'CLERK' ;
```

```
ENAME        JOB          DEPTNO
-----------------------------------------------
JAMES        CLERK          30
SMITH        CLERK          20
ADAMS        CLERK          20
```

- 위에서 **SELECT**문장은 업무가 **CLERK**인 모든 종업원의 이름 업무 그리고 부서 번호를 검색
- 업무 **CLERK**은 **EMP**테이블의 **job**열과 일치되도록 하기 위해 대문자로 명시 되어야 한다**.**

문자 스트링과 날짜

- 문자 스트링과 날짜 값은 단일 인용부호(‘’)로 둘러싸여 있다.
- 문자 값은 대소문자를 구분하고 날짜 값은 날짜 형식을 구분한다.
- 디폴트 날짜 형식은**‘DD-MON-YY’**이다

```
SQL>  SELECT     ename,   JOB,   deptno
2         FROM       emp
3         WHERE     ename = ‘JAMES’ ;
```

비교 연산자

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

구문 형식

```
WHREE expr Operator value
```

```
SQL> SELECT   ename,  sal,  comm
2       FROM      emp
3       WHERE     sal <= comm ;
```

```
ENAME          SAL          COMM
-----------------------------------------------
MARTIN         1250          1400
```

다른 비교 연산자

연산자	의미
BETWEEN....AND.....	두 값의 사이(포함하는)
IN(list)	어떤 값의 목록과 일치
LIKE	문자 패턴과 일치
IS NULL	Null 값

BETWEEN 연산자 사용

- 값의 범위에 해당하는 행을 디스플레이 할 수 있다. 명시한 범위는 하한 값과 상한 값을 포함한다. 아래의 **SELECT**문장은 급여가 **$1000**에서 **$1500**사이에 있는 종업원에 대해서 **EMP**테이블로부터 행을 리턴

```
SQL> SELECT   ename,  sal
2      FROM     emp
3      WHERE    sal  BETWEEN 1000 AND 1500 ;
```

ENAME	SAL
MARTIN	1250
TURNER	1500
WARD	1250
ADAMS	1100

IN 연산자 사용

- 명시된 목록에 있는 값에 대해서 테스트하기 위해. 아래의 예는 관리자의 종업원 번호가 **7902,7566,7788**인 모든 종업원의 종업원 번호, 이름, 급여 그리고 관리자의 종업원 번호를 디스플레이 한다.

```
SQL> SELECT   empno,  ename,  sal,   mgr
2       FROM      emp
3       WHERE     mgr  IN(7902,7566,7788);
```

```
EMPNO          ENAME          SAL          MGR
-------------------------------------------------------------------
7902            FORD          3000          7566
7369           SMITH           800          7902
7788           SCOTT          3000          7566
7876           ADAMS          1100          7788
```

LIKE 연산자 사용

- 검색 스트링 값에 대한 와일드 카드 검색을 위해서 **LIKE** 연산자를 사용한다.
- 검색 조건은 리터럴 문자나 숫자를 포함할 수 있다.

- %는 문자가 없거나 또는 하나 이상을 나타낸다.

- _는 하나의 문자를 나타낸다.

```
SQL>  SELECT    ename
2       FROM      emp
3       WHERE     ename   LIKE  'S%';
```

-> S로 시작하는 어떤 종업원에 대해서 EMP테이블로부터의 종업원 이름 리턴

LIKE 연산자 사용

- 패턴 일치 문자를 조합할 수 있다.

```
SQL>  SELECT    ename
2       FROM      emp
3       WHERE    ename   LIKE  '_A%';
```

```
ENAME
--------------------
JAMES
WARD
```

-> 이름의 두 번째 문자가 A인 모든 종업원의 이름을 디스플레이

- "%" 나 "_"에 대해 검색하기 위해 **ESCAPE**식별자를 사용할 수 있다.

IS NULL 연산자 사용

- **IS NULL 연산자로 null값을 테스트 할 수 있다.**

```
SQL>  SELECT    ename ,  mgr
2       FROM      emp
3       WHERE     mgr IS NULL;
```

```
ENAME      MGR
----------------------
KING
```

-> 관리자가 없는 모든 종업원의 이름과 관리자를 검색.

: null값은 값이 없거나 알 수 없거나 또는 적용할 수 없음을 의미한다. 그러므로 null값은 어떤 값과 같거나 또는 다를 수 없으므로 = 로는 테스트 할 수 없다.

논리 연산자

연산자	의미
AND	양쪽 컴포넌트의 조건이 true이면 true값을 리턴
OR	한쪽 컴포넌트의 조건만 true이면 true를 리턴
NOT	이후의 조건이 false이면 true를 리턴

AND연산자 사용

```
SQL> SELECT   empno,  ename,   job,  sal
2       FROM      emp
3       WHERE     sal >= 1100
4       AND       job = 'CLERK' ;
```

```
EMPNO       ENAME         JOB          SAL
-------------------------------------------------------------
7876            ADAMS       CLERK          1100
7934            MILLER      CLERK          1300
```

-> 양쪽의 조건이 참이어야 한다. 업무가 CLERK이고 급여가 $1100이상인 종업원이 선택됨

OR 연산자 사용

```
SQL> SELECT  empno,  ename,  job,  sal
2       FROM     emp
3       WHERE    sal >= 1100
4       OR           job = 'CLERK' ;
```

EMPNO	ENAME	JOB	SAL
7839	KING	PRESIDENT	5000
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7566	JONES	SALESMAN	1250

-> 한쪽의 조건만 참이면 된다. 업무가 CLERK이거나 급여가 $1100이상인 종업원이 선택됨

NOT 연산자 사용

```
SQL>  SELECT     ename ,   job
2         FROM         emp
3         WHERE       job  NOT  IN (‘CLERK’, ‘MANAGER’, ‘ANALYST’) ;
```

```
ENAME                JOB
-----------------------------------------
KING                 PRESIDENT
MARTIN               SALESMAN
ALLEN                SALESMAN
TURNER               SALESMAN
WARD                 SALESMAN
```

-> 업무가 CLERK, MANAGER 또는 ANALYST가 아닌 모든 종업원의 이름과 업무를 디스플레이 한다.

우선 순위 규칙

우선순위	연산자
1	모든 비교 연산자
2	NOT
3	AND
4	OR

우선 순위 규칙

```
SQL>  SELECT   ename,   job,  sal
2        FROM      emp
3        WHERE    job = 'SALESMAN'
4        OR         job = 'PRESIDENT'
5        AND       sal > 1500 ;
```

ENAME	JOB	SAL
KING	PRESIDENT	5000
MARTIN	SALESMAN	1250
ALLEN	SALESMAN	1600
TURNER	SALESMAN	1500
WARD	SALESMAN	1250

-> 업무가 PRESIDENT 이고 $1500 이상을 벌거나 또는 업무가 SALESMAN 인 행을 검색한다.

우선 순위 규칙

```
SQL> SELECT   ename,   job,  sal
2       FROM      emp
3       WHERE    ( job = 'SALESMAN'
4       OR          job = 'PRESIDENT')
5       AND        sal > 1500 ;
```

```
ENAME                JOB              SAL
-----------------------------------------------------
KING                 PRESIDENT        5000
ALLEN                SALESMAN         1600
```

-> 우선 순위를 강제로 바꾸기 위해서 괄호를 사용한다.

-> 업무가 PRESIDENT 이거나 SALESMAN이고 $1500 이상을 버는 행을 검색한다.

ORDER BY 절

- **ORDER BY** 절로 행을 정렬 한다.
  - **ASC :** 오름 차순, 디폴트
  - **DESC :** 내림 차순
- **ORDER BY** 절은 **SELECT** 문장의 가장 뒤에 온다.

```
SQL> SELECT     ename, job, deptno, hiredate
2    FROM       emp
3    ORDER BY   hiredate
```

```
ENAME      JOB          DEPTNO      HIREDATE
---------- ---------- ---------- ---------
SMITH      CLERK            20    17-DEC-80
ALLEN      SALESMAN         30    20-FEB-81
```

내림차순 정렬

```
SQL> SELECT     ename,  job,  deptno,  hiredate
2       FROM        emp
3       ORDER  BY  hiredate  DESC ;
```

ENAME	JOB	DEPTNO	HIREDATE
ADAMS	CLERK	20	12-JAN-83
SCOTT	ANALYST	20	09-DEC-82
MILLER	CLERK	10	23-JAN-82
JAMES	CLERK	30	03-DEC-81
FORD	ANALYST	20	03-DEC-81
KING	PRESIDENT	10	17-NOV-81
MARTIN	SALESMAN	30	28-SEP-81

열 별칭에 의한 정렬

```
SQL> SELECT    empno, ename, sal*12 annsal
2    FROM      emp
3    ORDER BY  annsal
```

EMPNO	ENAME	ANNSAL
7369	SMITH	9600
7900	JAMES	11400
7876	ADAMS	13200
7654	MARTIN	15000
7521	WARD	15000
7934	MILLER	15600

-> ORDER BY 절에서 열 별칭을 사용할 수 있다. 위는 연봉으로 데이터를 정렬

다중 열에 의한 정렬

•ORDER BY list에 명시한 순서가 정렬되는 순서

```
SQL> SELECT     ename, deptno,  sal
2    FROM       emp
3    ORDER  BY  deptno, sal  DESC ;
```

```
ENAME              DEPTNO        SAL
--------------------------------------------------
KING                 10           5000
CLARK                10           2450
MILLER               10           1300
FORD                 20           3000
```

-> 하나 이상의 열로 질의 결과를 정렬할 수 있다. 주어진 테이블에 있는 열의 개수 까지만 가능함. ORDER BY절에서 열을 명시하고 열 이름 뒤에 DESC를 명시한다. SELECT 절에 포함되지 않는 열로 정렬할 수도 있다.