

Chapter 3. 정렬과 연산

09강. 정렬 - ORDER BY

1) ORDER BY로 검색 결과 정렬하기

-- WHERE 구 뒤에 ORDERBY 구를 지정하는 경우
SELECT 열명 FROM 테이블명 WHERE 조건식 ORDER BY 열명
-- FROM 구 뒤에 ORDERBY 구를 지정하는 경우
SELECT 열명 FROM 테이블명 ORDER BY 열명

- 예제 3-1. age열의 값을 오름차순으로 정렬하기

```
mysql> SELECT * FROM sample31 ORDER BY age;
+-----+-----+-----+
| name | age | address          |
+-----+-----+-----+
| B씨  | 18  | 부산광역시 연제구 |
| C씨  | 25  | 서울특별시 중구   |
| A씨  | 36  | 대구광역시 중구   |
+-----+-----+-----+
3 rows in set (0.151 sec)
```

- 예제 3-2. address열로 정렬하기

```
mysql> SELECT * FROM sample31 ORDER BY address;
+-----+-----+-----+
| name | age | address          |
+-----+-----+-----+
| A씨  | 36  | 대구광역시 중구   |
| B씨  | 18  | 부산광역시 연제구 |
| C씨  | 25  | 서울특별시 중구   |
+-----+-----+-----+
3 rows in set (0.009 sec)
```

2) ORDER BY DESC로 내림차순으로 정렬하기

내림차순은 열명 뒤에 DESC를 붙여 지정

-- 내림차순으로 정렬

SELECT 열명 FROM 테이블명 ORDER BY 열명 DESC

-- 오름차순으로 정렬

SELECT 열명 FROM 테이블명 ORDER BY 열명 ASC

💡 정렬방법 지정 X → ASC로 간주

- 예제 3-3. DESC와 ASC

```
mysql> SELECT * FROM sample31 ORDER BY age DESC;
+-----+-----+-----+
| name | age | address          |
+-----+-----+-----+
| A씨  | 36 | 대구광역시 중구 |
| C씨  | 25 | 서울특별시 중구 |
| B씨  | 18 | 부산광역시 연제구 |
+-----+-----+-----+
3 rows in set (0.057 sec)

mysql> SELECT * FROM sample31 ORDER BY age ASC;
+-----+-----+-----+
| name | age | address          |
+-----+-----+-----+
| B씨  | 18 | 부산광역시 연제구 |
| C씨  | 25 | 서울특별시 중구 |
| A씨  | 36 | 대구광역시 중구 |
+-----+-----+-----+
3 rows in set (0.006 sec)
```

3) 대소관계

- 수치형 데이터: 숫자의 크기로 판별
- 날짜형 데이터: 숫자 크기로 판별 → 크다=최근
- 문자열형 데이터: 사전식 순서에 의해 결정 → 'ㄱ' < 'ㄴ'



사전식 순서에서 주의할 점

- 예제 3-5. sample311을 a열로 정렬하기

```
mysql> SELECT * FROM sample311 ORDER BY a;
```

a	b
1	1
10	10
11	11
2	2

a열은 문자열형으로, 대소관계를 사전식 순서로 비교함

즉, 문자열형 열에 숫자를 저장하면 문자로 인식되어 대소관계의 계산 방법이 달라짐

4) ORDER BY는 테이블에 영향을 주지 않는다

ORDER BY는 서버에서 클라이언트로 행 순서를 바꾸어 결과를 반환하는 것뿐, 저장장치에 저장된 데이터 행 순서까지 변경하진 않는다.

SELECT 명령은 데이터를 참조할 뿐, 변경은 하지 않는다.

10강. 복수의 열을 지정해 정렬하기

1) 복수 열로 정렬 지정

Q1. ORDER BY로 행을 정렬하지 않은 경우의 행의 순서는?

Q2. ORDER BY 구가 생략 됐을 때 순서는?

A1, A2 : 순서는 일정하지 않다!

⇒ 따라서 언제나 같은 순서로 결과를 얻고 싶다면 ORDER BY로 행 지정 필요

- 예제3-6. sample32테이블 내용 참조하기

```
mysql> SELECT * FROM sample32;
+-----+-----+
| a     | b     |
+-----+-----+
| 1     | 1     |
| 2     | 1     |
| 2     | 2     |
| 1     | 3     |
| 1     | 2     |
+-----+-----+
5 rows in set (0.131 sec)
```

- b는 a의 하위 번호
- 예제 3-7. sample32를 a열만으로 정렬하기

```
mysql> SELECT * FROM sample32 ORDER BY a;
+-----+-----+
| a     | b     |
+-----+-----+
| 1     | 1     |
| 1     | 3     |
| 1     | 2     |
| 2     | 1     |
| 2     | 2     |
+-----+-----+
5 rows in set (0.010 sec)
```

- b열은 정렬 순서 일정 X
→ a열에서의 똑같은 값에 대해 순서를 결정할 수 없기 때문

ORDER BY로 복수 열 지정하기

coma(,)로 열명을 구분해 지정하기

```
-- 복수 열로 정렬하기
SELECT 열명 FROM 테이블명 ORDER BY 열명1, 열명2,...
```

- 값이 같아 순서를 지정할 수 없는 경우, 다음으로 지정한 열명을 기준으로 정렬
- 예제 3-8. sample32를 a열과 b열로 정렬하기

```
mysql> SELECT * FROM sample32 ORDER BY a,b;
+-----+-----+
| a     | b     |
+-----+-----+
| 1     | 1     |
| 1     | 2     |
| 1     | 3     |
| 2     | 1     |
| 2     | 2     |
+-----+-----+
```

- 예제 3-9. sample32를 b열과 a열로 정렬하기

```
mysql> SELECT * FROM sample32 ORDER BY b,a;
+-----+-----+
| a     | b     |
+-----+-----+
| 1     | 1     |
| 2     | 1     |
| 1     | 2     |
| 2     | 2     |
| 1     | 3     |
+-----+-----+
```

2) 정렬방법 지정하기

복수 열을 지정하는 경우에도 각 열에 대한 정렬방법을 개별적으로 지정 가능하다

-- 복수 열 정렬

SELECT 열명 FROM 테이블명 ORDER BY 열명1[ASC/DESC], 열명2[ASC/DESC]...

- 예제 3-10. a열을 ASC, b열을 DESC로 정렬하기


```
mysql> SELECT * FROM sample32 ORDER BY a ASC, b DESC;
+-----+-----+
| a     | b     |
+-----+-----+
| 1     | 3     |
| 1     | 2     |
| 1     | 1     |
| 2     | 2     |
| 2     | 1     |
+-----+-----+
```

3) NULL 값의 정렬순서

- NULL의 특성상 대소비교가 어려움 ⇒ 가장 먼저 표시/가장 나중에 표시됨
- 데이터베이스 제품에 따라 기준이 다름
- MySQL의 경우, NULL값은 가장 작은 값으로 취급
 - ASC에서는 가장 먼저, DESC에서는 가장 나중에 표시

11강. 결과 행 제한하기 - LIMIT

1) 행수 제한

 LIMIT 구는 표준 SQL이 X → MySQL과 PostgreSQL에서 사용하는 문법
LIMIT구는 SELECT명령의 마지막에 지정(WHERE구/ORDER BY구 뒤에 지정)

```
-- LIMIT 구
SELECT 열명 FROM 테이블명 WHERE 조건식 ORDER BY 열명 LIMIT 행수
```

- LIMIT 다음에는 최대 행수를 수치로 지정
- 예제 3-12. LIMIT으로 행수 제한하기

```
mysql> SELECT * FROM sample33 LIMIT 3;
+-----+
| no    |
+-----+
|      1 |
|      2 |
|      3 |
+-----+
```

정렬한 후 제한하기

- 예제 3-13. 정렬 후 LIMIT로 행수 제한하기

```
mysql> SELECT * FROM sample33 ORDER BY no DESC LIMIT 3;
+-----+
| no    |
+-----+
|      7 |
|      6 |
|      5 |
+-----+
```

LIMIT를 사용할 수 없는 데이터베이스에서의 행 제한

- LIMIT은 표준 SQL이 아니기 때문에 MySQL과 PostgreSQL이외에서는 쓸 수 없음
- **SQL Server**에서는 비슷한 기능인 'TOP'이 있음

```
SELECT TOP 3 * FROM sample33;
```

- **Oracle**에는 LIMIT, TOP없음

ROWNUM이라는 열을 이용해 WHERE구로 조건 지정하여 행 제한

```
SELECT * FROM sample33 WHERE ROWNUM<=3;
```

2) 오프셋 지정

- 커뮤니티 사이트의 페이지 나누기 기능은 LIMIT을 이용해 구현
- 그 다음 페이지에서 6번째 행부터 표시
 - '6번째 행부터'는 결과값으로부터 데이터를 취득할 위치를 가리키는 것
→ LIMIT구의 OFFSET으로 지정
- 예제 3-14. LIMIT로 첫 번째 페이지에서 표시할 데이터 취득하기

```
mysql> SELECT * FROM sample33 LIMIT 3 OFFSET 0;
+-----+
| no    |
+-----+
|      1 |
|      2 |
|      3 |
+-----+
```

- OFFSET에 의한 시작 위치 조정은 LIMIT뒤에 기술

- 위치 지정은 0부터 시작하는 자료구조 배열의 인덱스를 생각
- 시작할 행 -1

-- OFFSET 지정

SELECT 열명 FROM 테이블명 LIMIT 행수 OFFSET 위치

- 예제 3-15. LIMIT로 두 번째 페이지에 표시할 데이터 취득하기

```
mysql> SELECT * FROM sample33 LIMIT 3 OFFSET 3;
+-----+
| no    |
+-----+
|      4 |
|      5 |
|      6 |
+-----+
```

12강. 수치 연산

1) 사칙 연산

산술연산자

연산자	연산
+	덧셈
-	뺄셈
*	곱셈
/	나눗셈
&	나머지

연산자의 우선순위

우선순위	연산자
1	* / %
2	+ -

2) SELECT 구로 연산하기

SELECT에서 열명 지정 뿐만 아니라 여러 가지 식(열명, 연산자, 상수로 구성) 기술도 가능

- 예제 3-17. SELECT구로 금액 계산하기

```
mysql> SELECT *, price*quantity FROM sample34;
```

no	price	quantity	price*quantity
1	100	10	1000
2	230	24	5520
3	1980	1	1980

3) 열의 별명

열이 길고 알아보기 어려운 경우는 별명을 붙여 열명을 재지정

- 예제 3-18. SELECT구에서 식에 별명 붙이기

```
mysql> SELECT *, price*quantity AS amount FROM sample34;
```

no	price	quantity	amount
1	100	10	1000
2	230	24	5520
3	1980	1	1980

AS: 별명 예약어

- 콤마(,)로 구분하여 복수의 식, 각각의 식에 지정 가능
- 기본적으로 중복되지 않게 설정
- 키워드 AS는 생략 가능
- 별명은 영어, 숫자, 한글 등으로 지정 가능
 - 한글로 지정 시, 더블쿼트(백쿼드)로 둘러싸서 지정 : ASCII 문자 외의 것을 사용할 때
 - 더블쿼트로 둘러싸면 db객체라고 간주

- 싱글쿼트로 둘러싸면 문자열 상수
- 더블쿼트 이용시 예약어와 같은 이름 사용 가능
- 숫자로 시작 불가능

4) WHERE 구에서 연산하기

- 예제 3-19. WHERE 구에서 계산, 검색하기

```
mysql> SELECT *, price*quantity AS amount FROM sample34
-> WHERE price*quantity >= 2000;
+-----+-----+-----+-----+
| no    | price | quantity | amount |
+-----+-----+-----+-----+
|      2 |    230 |        24 |    5520 |
+-----+-----+-----+-----+
```

- WHERE 구에 amount 별명으로 지정 시, 에러 발생

WHERE구와 SELECT구의 내부처리 순서

데이터 서버 내부에서 **WHERE 구 → SELECT 구**의 순서로 처리

⇒ WHERE 구의 처리는 SELECT 구보다 선행됨

⇒ **SELECT 구에서 지정한 별명은 WHERE 구 안에서 사용할 수 없다!**

5) NULL 값의 연산

NULL로 연산하면 결과는 NULL

6) ORDER BY 구에서 연산하기

- 예제 3-20. 금액을 내림차순으로 정렬하기

```
mysql> SELECT *, price*quantity AS amount FROM sample34
-> ORDER BY price*quantity DESC;
```

no	price	quantity	amount
2	230	24	5520
3	1980	1	1980
1	100	10	1000

- ORDER BY는 서버에서 내부적으로 가장 나중에 처리 됨
⇒ SELECT에서 지정한 별명을 ORDER BY에서 사용가능
- 예제 3-21. 금액의 별명을 사용해 내림차순으로 정렬하기

```
mysql> SELECT *, price*quantity AS amount FROM sample34
-> ORDER BY amount DESC;
```

no	price	quantity	amount
2	230	24	5520
3	1980	1	1980
1	100	10	1000

 WHERE 구 → SELECT 구 → ORDER BY 구

7) 함수

-- 함수

함수명 (인수1, 인수2, ...)

- 함수명에 따라 연산 방법 결정
- 계산 대상을 인수(파라미터)로 지정
- 함수의 반환값 : 연산 결과값 반환
- 함수도 연산자도 표기방법이 다를 뿐, 같은 것임

8) ROUND 함수

ROUND 함수는 반올림을 하는 함수

- 예제 3-22. ROUND로 반올림하기

```
mysql> SELECT amount, ROUND(amount) FROM sample341;
```

amount	ROUND(amount)
5961.60	5962
2138.40	2138
1080.00	1080

- DECIMAL 형: 정수부와 소수부의 자릿수를 지정할 수 있는 자료형(소수점을 포함하는 수치)

반올림 자릿수 지정

- ROUND는 함수는 기본적으로 소수점 첫째 자리에서 반올림
- ROUND 함수의 두 번째 인수는 반올림할 자릿수 지정
- 예제 3-23. 소수점 둘째자리에서 반올림하기

```
mysql> SELECT amount, ROUND(amount, 1) FROM sample341;
```

amount	ROUND(amount, 1)
5961.60	5961.6
2138.40	2138.4
1080.00	1080.0

- 음수로 지정하여 정수부의 반올림할 자릿수 지정 가능
 - -1은 1단위, -2는 10단위...
- TRUNCATE 함수: 버림 함수
- 예제 3-24. 10단위를 반올림하기

```
mysql> SELECT amount, ROUND(amount, -2) FROM sample341;
```

amount	ROUND(amount, -2)
5961.60	6000
2138.40	2100
1080.00	1100

13강. 문자열 연산

1) 문자열 결합

문자열 결합은 문자열 데이터를 결합하는 연산이다

- 문자열 결합 연산자(데이터베이스 제품마다 차이가 있음)

연산자/함수	연산	데이터베이스
+	문자열 결합	SQL Server
	문자열 결합	Oracle, DB2, PostgreSQL
CONCAT	문자열 결합	MySQL

- 예제 3-25. 문자열 결합

```
mysql> SELECT CONCAT(quantity, unit) FROM sample35;
+-----+
| CONCAT(quantity, unit) |
+-----+
| 10개                  |
| 24통                  |
| 1장                   |
+-----+
```

- unit은 문자열형, quantity는 인티저형이지만 CONCAT함수는 수치 데이터도 연산 가능
but, 문자열로 결합한 결과는 문자열형이 된다

2) SUBSTRING 함수

문자열의 일부분을 계산해서 반환해주는 함수이다

- 사용되는 예시
 - 블로그 시스템 내의 ID
 - 연월일(YYYYMMDD)과 같은 날짜 데이터에서 따로 추출할 때
 - 앞자리 4자리(년 추출)

SUBSTRING('20140125001', 1, 4) → '2014'

- 5째 자리부터 2자리(월 추출)

SUBSTRING('20140125001', 5, 2) → '01'

3) TRIM 함수

문자열의 앞뒤로 여분의 스페이스가 있을 경우 이를 제거하는 함수이다

(문자열 도중에 존재하는 스페이스는 제거 X)

- 고정길이 문자열형에 대해 많이 사용
- TRIM으로 스페이스 제거하기

TRIM('ABC ') → 'ABC'

4) CHARACTER_LENGTH 함수

문자열의 길이를 계산해서 돌려주는 함수이다

- CHAR_LENGTH로 줄여 사용 가능
- OCTET_LENGTH : 문자열의 길이를 바이트 단위로 계산하여 돌려주는 함수
- 문자열 데이터도 수치로 저장됨
- 문자를 수치화(인코드)하는 방식에 따라 필요한 저장공간 크기가 달라짐

문자세트 : 한 문자가 몇 바이트인지는 쓰이는 문자세트에 따라 다르다

- 반각 문자는 전각문자 폭의 절반, 저장용량도 더 작음
- 'ASCII문자' : 반각의 알파벳이나 숫자, 기호
- 문자세트: 인코드 방식이 데이터베이스나 테이블을 정의할 때 변경가능

14강. 날짜 연산

1) SQL에서의 날짜

- 날짜나 시간 데이터는 수치 데이터와 같이 사칙 연산 가능

- 기간형 데이터: 기간(간격)의 차를 나타내는 자료형 (10일간, 2시간 10분 등)

시스템 날짜

하드웨어 상의 시계로부터 실시간으로 얻을 수 있는 일시적인 데이터

- 'CURRENT_TIMESTAMP' : 시스템 날짜 시간 실시간 확인 함수
 - 함수이지만 인수 필요 X → 괄호 사용X
- 예제 3-26. 시스템 날짜 확인하기

```
mysql> SELECT CURRENT_TIMESTAMP;
+-----+
| CURRENT_TIMESTAMP |
+-----+
| 2025-09-14 16:41:34 |
+-----+
```

- 옛 FROM구 생략가능?!
SELECT구는 SELECT명령으로 실행 가능 but, Oracle은 생략 불가능

날짜 서식

- 날짜 서식은 국가별로 다름
- 임의의 날짜를 저장하고 싶음
⇒ 날짜 데이터의 서식을 임의로 지정, 변환할 수 있는 함수 지원
 - Oracle의 경우, TO_DATE함수
TO_DATE('2014/01/25', 'YYYY/MM/DD')
 - 날짜형 데이터를 문자열 데이터로 출력하는 함수 : Oracle의 TO_CHAR

2) 날짜의 덧셈과 뺄셈

날짜시간형 데이터는 기간형 수치데이터와 덧셈 및 뺄셈 가능

- 연산 후 날짜시간형 데이터가 반환
예) a + 1 DAY(특정일로부터 1일 후)
- 예제 3-27. 시스템 날짜의 1일 후를 계산하기

```
mysql> SELECT CURRENT_DATE + INTERVAL 1 DAY;
+-----+
| CURRENT_DATE + INTERVAL 1 DAY |
+-----+
| 2025-09-15                      |
+-----+
```

- CURRENT_DATE : 시스템 날짜의 날짜만 확인하는 함수
- INTERVAL 1 DAY : '1일 후'라는 의미의 기간형 상수

날짜형 간의 뺄셈(/덧셈)

- Oracle : '2014-02-28' - '2014-01-01'
- MySQL : DATEDIFF('2014-02-28', '2014-01-01')

15강. CASE 문으로 데이터 변환하기

임의의 조건에 따라 독자적으로 변환 처리를 지정해 데이터를 변환하고 싶을 때 사용

1) CASE 문

- 간단한 처리의 경우 사용자 정의 함수 작성하지 않고 CASE문으로 처리 가능

```
-- CASE문
CASE WHEN 조건식 1 THEN 식1
[ WHEN 조건식2 THEN 식2...]
[ ELSE 식3 ]
END
```

- WHEN절에 참과 거짓을 반환하는 조건식 기술
- 참일시 THEN절의 식이 처리
- 그 어떤 조건식도 만족하지 못할 시 ELSE절에 기술한 식 채택
 - ELSE 생략 시, 'ELSE NULL'로 간주
- 예제 3-28. NULL 값을 0으로 변환하기


```
mysql> SELECT a, CASE WHEN a IS NULL THEN 0 ELSE a END "a(null=0)"
-> FROM sample37;
```

a	a(null=0)
1	1
2	2
NULL	0

COALESCE

NULL값을 변환하는 함수

- COALESCE 함수를 이용해 예제 3-28. 재작성

```
SELECT a, COALESCE(a, 0) FROM sample37;
```

2) 또 하나의 CASE 문

- 디코드 : 문자화(수치데이터→문자형데이터)
- 인코드: 수치화(문자형데이터→수치데이터)

```
WHEN a =1 THEN '남자'
WHEN b=1 THEN '여자'
```

CASE문에는 두 개의 구문이 있다

- 검색 CASE : 'CASE WHEN 조건식 THEN식...'
- 단순 CASE : 'CASE WHEN 식 THEN 식...' → WHEN 뒤에 조건식 X, 그냥 식

```
-- 단순 CASE 식
CASE 식1
  WHEN 식2 THEN 식3
  [ WHEN 식4 THEN 식5...]
  [ ELSE 식6 ]
END
```

- 예제 3-29. 성별 코드 변환하기(검색 CASE)

```
mysql> SELECT a AS "코드",
-> CASE
-> WHEN a=1 THEN '남자'
-> WHEN a=2 THEN '여자'
-> ELSE '미지정'
-> END AS "성별" FROM sample37;
```

코드	성별
1	남자
2	여자
NULL	미지정

- 예제 3-30. 성별 코드 변환하기(단순 CASE)

```
mysql> SELECT a AS "코드",
-> CASE a
-> WHEN 1 THEN '남자'
-> WHEN 2 THEN '여자'
-> ELSE '미지정'
-> END AS "성별" FROM sample37;
```

코드	성별
1	남자
2	여자
NULL	미지정

3) CASE를 사용할 경우 주의사항

- CASE문은 SELECT 뿐만 아니라 어디서든 사용 가능 !

ELSE 생략

- ELSE를 생략하면 ELSE NULL이 됨!
⇒ CASE문의 ELSE는 생략하지 않는 편이 좋다

WHEN에서 NULL 지정하기

```
-- 예제. 단순 CASE 문에서 WHEN 절에 NULL지정하기
CASE a
```

```

WHEN 1 THEN '남자'
WHEN 2 THEN '여자'
WHEN NULL THEN '데이터 없음'
ELSE '미지정'
END

```

- 예제 조건식 처리 순서

1. a=1
2. a=2
3. a=NULL

→ 그러나 비교연산자 = 으로 NULL을 비교할 수 없음

→ 따라서 a 열이 NULL 값이더라도 '미지정'으로 반환됨

⇒ **IS NULL** 사용

- 예제 3-30. 검색 CASE문으로 NULL 판정하기

```

mysql> SELECT a AS "코드",
-> CASE a
-> WHEN 1 THEN '남자'
-> WHEN 2 THEN '여자'
-> ELSE '미지정'
-> END AS "성별" FROM sample37;

```

코드	성별
1	남자
2	여자
NULL	미지정

DECODE NVL

- Oracle: 디코드를 수행하는 DECODE함수
 - but 다른 제품에서는 지원 안 하므로 보통 CASE를 많이 씀
- NULL값 반환 함수
 - Oracle : NVL
 - SQL Server : ISNULL

- 표준 SQL: COALESCE