7장 데이터 검색 명령문

1. 서론
2. SELECT 명령문 모든 절을 포함한 수행 과정
3. SELECT 명령문 일부 절을 포함한 수행 과정

7.1 서론

● SELECT 명령문은 최대 6개 절(Clause)로 구성

```
<select statement> ::=
      <select clause>
      <from clause>
      [ <where clause> ]
      [ <group by clause>
       [<having clause>] ]
      [ <order by clause> ]
```

● SELECT 명령문을 구성할 때 규칙 및 주의할 점

-SELECT 명령문은 최소 2개 절(SELECT, FROM)로 구성
-WHERE, GROUP BY, ORDER BY 같은 절은 선택적으로 사용
-절의 순서는 고정(GROUP BY절은 WHERE 또는 FROM절 앞에 올 수 없다)
-ORDER BY절이 사용된다면 이 절은 항상 가장 나중에 사용
-HAVING절은 GROUP BY 절이 사용되어야 만이 사용 가능

- 정확한 SELECT 명령문의 사용의 예제

```
SELECT   ...
FROM     ...
ORDER BY ...
```

```
SELECT   ...
FROM     ...
GROUP BY ...
HAVING   ...
```

```
SELECT   ...
FROM     ...
WHERE    ...
```

7.2 SELECT 명령문 모든 절을 포함한 수행 과정

[예제 7-1] 등록 테이블("FEE")에서 장학금을 지급 받은 학생의 학번과 장학금 내역을 출력하라.

```
mysql> select stu_no, jang_total
    -> from fee
    -> where jang_total > 0;
```

[예제 7-2] 등록 테이블("FEE")에서 장학금을 1,000,000 이상 지급 받은 학생 중에서 2회 이상 지급받은 학생의 학번과 , 지급받은 횟수를 학번 내림차순으로 출력하라.

```
mysql> select stu_no, count(*)
    -> from fee
    -> where jang_total > 1000000
    -> group by stu_no
    -> having count(*) > 1
    -> order by stu_no desc;
+----------+----------+
| stu_no   | count(*) |
+----------+----------+
| 20161001 |        4 |
| 20141001 |        5 |
+----------+----------+
```

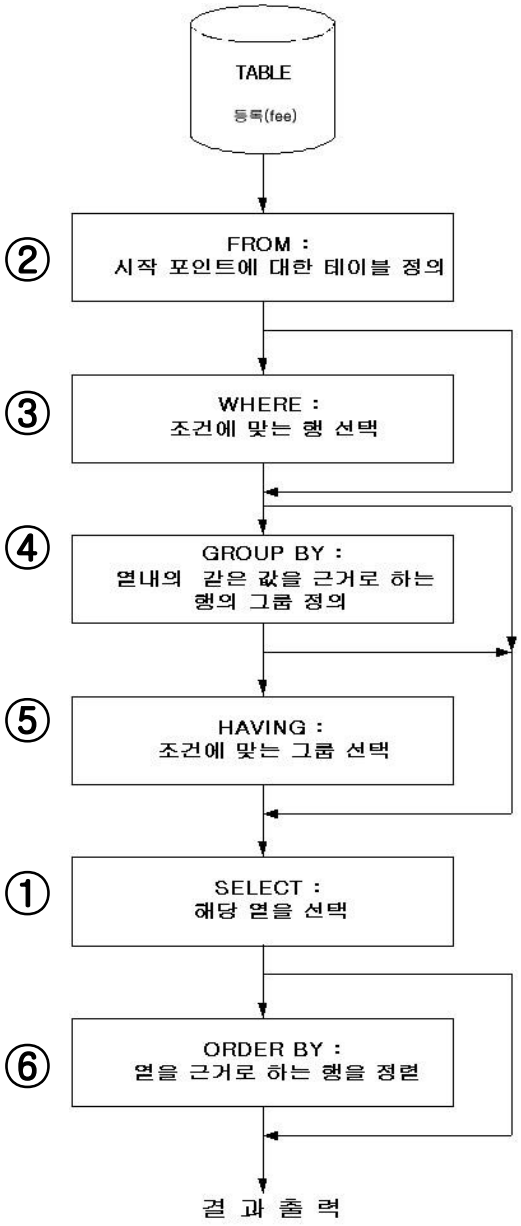
위의 SELECT 명령문은 장학금이 1,000,000원 이상을 받은 학생의 학번을 찾아서 2회 이상 장학금을 지급받은 학생의 결과를 학번 내림차순으로 출력한다.

● 각 절이 수행되는 순서

2번(FROM절)→3번(WHERE절)→4번(GROUP BY절)→5번(HAVING절)
→1번(SELECT절)→6번(ORDER BY절) 순으로 진행

◆ Select 명령문 6개 절 수행 순서

```
① select stu_no, count(*)
② from fee
③ where jang_total > 1000000
④ group by stu_no
⑤ having count(*) > 1
⑥ order by stu_no desc;
```



7.2.1 FROM 절

```
mysql> select * from fee;
```

stu_no	fee_year	fee_term	fee_enter	fee_price	fee_total	jang_code	jang_total	fee_pay	fee_div	fee_date
20141001	2014	1	500000	3000000	3500000	1	500000	3000000	Y	2014-02-18
20141001	2014	2	NULL	3000000	3000000	10	2500000	500000	Y	2014-08-20
20141001	2015	1	NULL	3000000	3000000	11	2000000	1000000	Y	2015-02-18
20141001	2015	2	NULL	3000000	3000000	21	800000	2200000	Y	2015-08-10
20141001	2018	1	500000	2500000	3000000	2	1000000	2000000	Y	2018-02-01
20141001	2018	2	NULL	2500000	2500000	10	2500000	0	Y	2018-08-10
20141001	2019	1	NULL	2800000	2800000	10	2500000	300000	Y	2019-02-15
20141001	2019	2	NULL	2800000	2800000	10	2500000	300000	Y	2019-08-16
20161001	2016	1	NULL	3000000	3000000	10	2500000	500000	Y	2016-02-14
20161001	2016	2	NULL	3000000	3000000	10	2500000	500000	Y	2016-08-18
20161001	2019	1	NULL	3000000	3000000	11	2000000	1000000	Y	2019-02-10
20161001	2019	2	NULL	3000000	3000000	10	2500000	500000	Y	2019-08-19
20191004	2019	1	500000	3000000	3500000	1	500000	3000000	Y	2019-02-18
20191004	2019	2	NULL	3000000	3000000	11	2000000	1000000	Y	2019-08-10
20191005	2019	1	500000	3000000	3500000	1	500000	3000000	Y	2019-02-18
20191005	2019	2	NULL	3000000	3000000	NULL	NULL	3000000	Y	2019-08-10
20191006	2019	1	500000	3000000	3500000	1	500000	3000000	Y	2019-02-18
20191006	2019	2	NULL	3000000	3000000	NULL	NULL	3000000	Y	2019-08-10
20191007	2019	1	500000	3000000	3500000	1	500000	3000000	Y	2019-02-18
20191007	2019	2	NULL	3000000	3000000	NULL	NULL	3000000	Y	2019-08-10
20191008	2019	1	500000	3000000	3500000	1	500000	3000000	Y	2019-02-18
20191008	2019	2	NULL	3000000	3000000	NULL	NULL	3000000	Y	2019-08-10
20201002	2020	1	500000	3000000	3500000	1	500000	3000000	Y	2020-02-18
20201002	2020	2	NULL	3000000	3000000	10	2500000	500000	Y	2020-08-10

24 rows in set (0.00 sec)

7.2.2 WHERE 절

JANG_TOTAL 열에 있는 값이 1,000,000원 이상이 되는 모든 행 선택하여 출력

```
mysql> select *
    -> from fee
    -> where jang_total > 1000000;
```

stu_no	fee_year	fee_term	fee_enter	fee_price	fee_total	jang_code	jang_total	fee_pay	fee_div	fee_date
20141001	2014	2	NULL	3000000	3000000	10	2500000	500000	Y	2014-08-20
20141001	2015	1	NULL	3000000	3000000	11	2000000	1000000	Y	2015-02-18
20141001	2018	2	NULL	2500000	2500000	10	2500000	0	Y	2018-08-10
20141001	2019	1	NULL	2800000	2800000	10	2500000	300000	Y	2019-02-15
20141001	2019	2	NULL	2800000	2800000	10	2500000	300000	Y	2019-08-16
20161001	2016	1	NULL	3000000	3000000	10	2500000	500000	Y	2016-02-14
20161001	2016	2	NULL	3000000	3000000	10	2500000	500000	Y	2016-08-18
20161001	2019	1	NULL	3000000	3000000	11	2000000	1000000	Y	2019-02-10
20161001	2019	2	NULL	3000000	3000000	10	2500000	500000	Y	2019-08-19
20191004	2019	2	NULL	3000000	3000000	11	2000000	1000000	Y	2019-08-10
20201002	2020	2	NULL	3000000	3000000	10	2500000	500000	Y	2020-08-10

```
11 rows in set (0.00 sec)
```

7.2.3 GROUP BY 절

-GROUP BY 절은 그룹별로 검색을 할 때 사용

-대표적인 그룹 함수 : COUNT( ), AVG( ), MIN( ), MAX( ), SUM( )
COUNT( )는 행이나 열의 개수를 구하는 함수, AVG( )는 평균,
MIN( )은 최소값, MAX( )는 최대값, SUM( )은 총 합을 구하는 함수

-학생별로 그룹을 만들려면 group by stu_no(학번)으로 표현

```
mysql> select stu_no, count(*)
    -> from fee
    -> where jang_total > 1000000
    -> group by stu_no;
+----------+----------+
| stu_no   | count(*) |
+----------+----------+
| 20141001 |        5 |
| 20161001 |        4 |
| 20191004 |        1 |
| 20201002 |        1 |
+----------+----------+
4 rows in set (0.00 sec)
```

7.2.4 HAVING 절

-GROUP BY 절의 조건절로 WHERE절 처럼 사용 가능

-대표적인 그룹 함수 COUNT( ), AVG( ), MIN( ), MAX( ), SUM( ) 등을 이용 HAVING 절(조건절)을 만듬

-두 번 이상의 장학금을 받은 학생만을 선택

```
mysql> select stu_no, count(*)
    -> from fee
    -> where jang_total > 1000000
    -> group by stu_no
    -> having count(*) > 1;
+----------+----------+
| stu_no   | count(*) |
+----------+----------+
| 20141001 |        5 |
| 20161001 |        4 |
+----------+----------+
2 rows in set (0.00 sec)
```

7.2.5 SELECT 절

SELECT 절은 최종 결과 테이블에 표현될 열을 지정하기 위해서 사용

7.2.6 ORDER BY 절

-ORDER BY 절은 마지막으로 수행되는 절로 선택된 행을 정렬
-SELECT 절에서 나온 결과 값은 입력된 데이터 순으로 출력

-최종 결과에서 학번 내림차순으로 정렬

```
mysql> select stu_no, count(*)
    -> from fee
    -> where jang_total > 1000000
    -> group by stu_no
    -> having count(*) > 1
    -> order by stu_no desc;
+----------+----------+
| stu_no   | count(*) |
+----------+----------+
| 20161001 |        4 |
| 20141001 |        5 |
+----------+----------+
2 rows in set (0.00 sec)
```

7.3 SELECT 명령문 일부 절을 포함한 수행 과정

[예제 7-3] 수강신청 테이블(ATTEND)에서 2006년도 1학기에 수강 신청한 학생의 학번과 수강년도, 학기, 교과목코드, 교수코드를 교수코드 오름차순으로 나타내어라.

```
mysql> select stu_no, att_year, att_term, sub_code, prof_code
    -> from attend
    -> where att_year = '2016' and att_term = 1
    -> order by prof_code;
+----------+----------+----------+----------+-----------+
| stu_no   | att_year | att_term | sub_code | prof_code |
+----------+----------+----------+----------+-----------+
| 20161001 | 2016     |        1 | 4004     | 4001      |
| 20161001 | 2016     |        1 | 4001     | 4002      |
| 20161001 | 2016     |        1 | 4002     | 4003      |
| 20161001 | 2016     |        1 | 4003     | 4004      |
| 20161001 | 2016     |        1 | 4005     | 4007      |
| 20161001 | 2016     |        1 | 4006     | 4008      |
+----------+----------+----------+----------+-----------+
6 rows in set (0.02 sec)
```

7.3.1 FROM 절

FROM 절에서는 수강(ATTEND) 테이블의 모든 행과 열을 출력한다.

stu_no	att_year	att_term	att_isu	sub_code	prof_code	att_point	att_grade	att_div	att_jae	att_date
20141001	2014	1	3	4001	4002	3	99	Y	1	2014-03-05
20141001	2014	1	4	4002	4003	3	95	Y	1	2014-03-05
20141001	2014	1	4	4003	4004	3	97	Y	1	2014-03-05
20141001	2014	1	4	4004	4001	3	98	Y	1	2014-03-05
20141001	2014	1	4	4005	4007	3	96	Y	1	2014-03-05
20141001	2014	1	4	4006	4008	3	95	Y	1	2014-03-05
20141001	2014	2	3	4007	4009	3	93	Y	1	2014-09-03
20141001	2014	2	4	4008	4005	3	92	Y	1	2014-09-03
20141001	2014	2	4	4009	4006	3	94	Y	1	2014-09-03
20141001	2014	2	4	4010	4001	3	90	Y	1	2014-09-03
20141001	2014	2	4	4011	4002	3	91	Y	1	2014-09-03
20141001	2014	2	4	4012	4003	3	92	Y	1	2014-09-03
20161001	2016	1	3	4001	4002	3	99	Y	1	2016-03-05
20161001	2016	1	4	4002	4003	3	95	Y	1	2016-03-05
20161001	2016	1	4	4003	4004	3	97	Y	1	2016-03-05
20161001	2016	1	4	4004	4001	3	98	Y	1	2016-03-05
20161001	2016	1	4	4005	4007	3	93	Y	1	2016-03-05
20161001	2016	1	4	4006	4008	3	95	Y	1	2016-03-05

18 rows in set (0.00 sec)

7.3.2 WHERE 절

조건으로 수강년도는 2006년, 수강학기는 1학기에 해당하는 두 가지 조건 (ATT_YEAR = '2006' AND ATT_TERM = 1)이 모두 만족한 행이 결과로 출력

stu_no	att_year	att_term	att_isu	sub_code	prof_code	att_point	att_grade	att_div	att_jae	att_date
20161001	2016	1	3	4001	4002	3	99	Y	1	2016-03-05
20161001	2016	1	4	4002	4003	3	95	Y	1	2016-03-05
20161001	2016	1	4	4003	4004	3	97	Y	1	2016-03-05
20161001	2016	1	4	4004	4001	3	98	Y	1	2016-03-05
20161001	2016	1	4	4005	4007	3	93	Y	1	2016-03-05
20161001	2016	1	4	4006	4008	3	95	Y	1	2016-03-05

7.3.3 GROUP BY 절

GROUP BY 절을 사용하지 않았으므로 중간 결과는 변하지 않고 그대로 유지

7.3.4 HAVING 절

HAVING 절을 사용하지 않았으므로 중간 결과는 변하지 않고 그대로 유지

7.3.5 SELECT 절

SELECT 절에서 STU_NO, ATT_YEAR, ATT_TERM, SUB_CODE, PROF_CODE 열을 지정했으므로 중간 결과는 다음과 같다.

stu_no	att_year	att_term	sub_code	prof_code
20161001	2016	1	4001	4002
20161001	2016	1	4002	4003
20161001	2016	1	4003	4004
20161001	2016	1	4004	4001
20161001	2016	1	4005	4007
20161001	2016	1	4006	4008

7.3.6 ORDER BY 절

ORDER BY PROF_CODE는 오름차순 정렬인 ASC(ENDING)이 생략된 경우로 교수코드를 우선적으로 오름차순 정렬하여 결과 값을 출력

stu_no	att_year	att_term	sub_code	prof_code
20161001	2016	1	4004	4001
20161001	2016	1	4001	4002
20161001	2016	1	4002	4003
20161001	2016	1	4003	4004
20161001	2016	1	4005	4007
20161001	2016	1	4006	4008