핵심정리

테이블에 계층형 데이터가 존재하는 경우 데이터를 조회하기 위해서 **계층형 질의(Hierarchical Query)**를 사용한다. 계층형 데이터란 동일 테이블에 계층적으로 상위와 하위 데이터가 포함된 데이터를 말한다. 예를 들어, 사원 테이블에서는 사원들 사이에 상위 사원(관리자)과 하위 사원 관계가 존재하고 조직 테이블에서는 조직들 사이에 상위 조직과 하위 조직 관계가 존재한다.

90 **다음 중 계층형 질의문에 대한 설명으로 가장 부적절한 것은?**

① SQL Server에서의 계층형 질의문은 CTE(Common Table Expression)를 재귀 호출함으로써 계층 구조를 전개한다.

② SQL Server에서의 계층형 질의문은 앵커 멤버를 실행하여 기본 결과 집합을 만들고 이후 재귀 멤버를 지속적으로 실행한다.

③ 오라클의 계층형 질의문에서 WHERE 절은 모든 전개를 진행한 이후 필터 조건으로서 조건을 만족하는 데이터만을 추출하는데 활용된다.

④ 오라클의 계층형 질의문에서 PRIOR 키워드는 CONNECT BY 절에만 사용할 수 있으며 'PRIOR 자식 = 부모' 형태로 사용하면 순방향 전개로 수행 된다.

91 **아래 [부서]와 [매출] 테이블에 대해서 SQL 문장을 실행하여 아래 [결과]와 같이 데이터가 추출 되었다. 다음 중 동일한 결과를 추출하는 SQL 문장은?**

아 래

[테이블 : 부서]

부서코드(PK)	부서명	상위 부서코드(FK)
100	아시아부	NULL
110	한국지사	100
111	서울지점	110
112	부산지점	110
120	일본지사	100
121	도쿄지점	120
122	오사카지점	120
130	중국지사	100
131	베이징지점	130
132	상하이지점	130
200	남유럽지부	NULL
210	스페인지사	200
211	마드리드지점	210
212	그라나다지점	210
220	포르투갈지사	200
221	리스본지점	220
222	포르투지점	220

[테이블 : 매출]

부서코드	매출액
111	1000
112	2000
121	1500
122	1000
131	1500
132	2000
211	2000
212	1500
221	1000
222	2000

[결과]

부서코드	부서명	상위부서코드	매출액	LVL
100	아시아지부	NULL	NULL	2
120	일본지사	100	NULL	1
121	도쿄지점	120	1500	2
122	오사카지점	120	1000	2

① SELECT A.부서코드, A.부서명, A.상위부서코드, B.매출액, LVL
FROM (SELECT 부서코드, 부서명, 상위부서코드, LEVEL AS LVL
FROM 부서
START WITH 부서코드 = '120'
CONNECT BY PRIOR 상위부서코드 = 부서코드
UNION
SELECT 부서코드, 부서명, 상위부서코드, LEVEL AS LVL
FROM 부서
START WITH 부서코드 = '120'
CONNECT BY 상위부서코드 = PRIOR 부서코드) A LEFT
OUTER JOIN 매출 B
ON (A.부서코드 = B.부서코드)
ORDER BY A.부서코드;

② SELECT A.부서코드, A.부서명, A.상위부서코드, B.매출액, LVL
FROM (SELECT 부서코드, 부서명, 상위부서코드, LEVEL AS LVL
FROM 부서
START WITH 부서코드 = '100'
CONNECT BY 상위부서코드 = PRIOR 부서코드) A LEFT
OUTER JOIN 매출 B
ON (A.부서코드 = B.부서코드)
ORDER BY A.부서코드;

③ SELECT A.부서코드, A.부서명, A.상위부서코드, B.매출액, LVL
FROM (SELECT 부서코드, 부서명, 상위부서코드, LEVEL AS LVL
FROM 부서
START WITH 부서코드 = '121'
CONNECT BY PRIOR 상위부서코드 = 부서코드) A LEFT
OUTER JOIN 매출 B
ON (A.부서코드 = B.부서코드)
ORDER BY A.부서코드;

④ SELECT A.부서코드, A.부서명, A.상위부서코드, B.매출액, LVL
FROM (SELECT 부서코드, 부서명, 상위부서코드, LEVEL AS LVL
FROM 부서
START WITH 부서코드 = (SELECT 부서코드
FROM 부서
WHERE 상위부서코드 IS
NULL
START WITH 부서코드
= '120'
CONNECT BY PRIOR
상위부서코드 = 부서
코드)
CONNECT BY 상위부서코드 = PRIOR 부서코드) A LEFT
OUTER JOIN 매출 B
ON (A.부서코드 = B.부서코드)
ORDER BY A.부서코드;

핵심정리

셀프 조인(Self Join)이란 동일 테이블 사이의 조인을 말한다. 따라서 FROM 절에 동일 테이블이 두 번 이상 나타난다. 동일 테이블 사이의 조인을 수행하면 테이블과 칼럼 이름이 모두 동일하기 때문에 식별을 위해 반드시 테이블 별칭(Alias)를 사용해야 한다.

셀프 조인(Self Join) 문장

```
SELECT
    ALIAS명1.칼럼명,
    ALIAS명2.칼럼명, ...
FROM
    테이블 ALIAS명1,
    테이블 ALIAS명2
WHERE
    ALIAS명1.칼럼명2 =
    ALIAS명2.칼럼명1;
```

92 다음 중 SELF JOIN을 수행해야 할 경우로 가장 적절한 것은?

① 한 테이블 내에서 두 칼럼이 연관 관계가 있다.
② 두 테이블에 연관된 칼럼은 없으나 JOIN을 해야 한다.
③ 두 테이블에 공통 칼럼이 존재하고 두 테이블이 연관 관계가 있다.
④ 한 테이블 내에서 연관된 칼럼은 없으나 JOIN을 해야 한다.

93 아래와 같이 일자별매출 테이블이 존재할 때 아래 결과처럼 일자별 누적매출액을 SQL로 구하려고 한다. WINDOW FUNCTION을 사용하지 않고 일자별 누적매출액을 구하는 SQL로 옳은 것은?

아 래

[테이블 : 일자별매출]

일자	매출액
2015.11.01	1000
2015.11.02	1000
2015.11.03	1000
2015.11.04	1000
2015.11.05	1000
2015.11.06	1000
2015.11.07	1000
2015.11.08	1000
2015.11.09	1000
2015.11.10	1000

[결과 : 일자별 누적매출액]

일자	누적매출액
2015.11.01	1000
2015.11.02	2000
2015.11.03	3000
2015.11.04	4000
2015.11.05	5000
2015.11.06	6000
2015.11.07	7000
2015.11.08	8000
2015.11.09	9000
2015.11.10	10000

①
```
SELECT A.일자, SUM(A.매출액) AS 누적매출액
FROM   일자별매출 A
GROUP BY A.일자
ORDER BY A.일자;
```

②
```
SELECT B.일자, SUM(B.매출액) AS 누적매출액
FROM   일자별매출 A JOIN 일자별매출 B ON (A.일자 >= B.일자)
GROUP BY B.일자
ORDER BY B.일자;
```

③
```
SELECT A.일자, SUM(B.매출액) AS 누적매출액
FROM   일자별매출 A JOIN 일자별매출 B ON (A.일자 >= B.일자)
GROUP BY A.일자
ORDER BY A.일자;
```

④
```
SELECT  A.일자
       ,(SELECT SUM(B.매출액)
         FROM   일자별매출 B WHERE B.일자 >= A.일자) AS 누적
매출액
FROM   일자별매출 A
GROUP BY A.일자
ORDER BY A.일자;
```

94 다음 중 아래의 SQL 수행 결과로 가장 적절한 것은?

아 래

```
SELECT COUNT(DISTINCT A||B)
FROM   EMP
WHERE  D = (SELECT D FROM DEPT WHERE E = 'i');
```

EMP 테이블

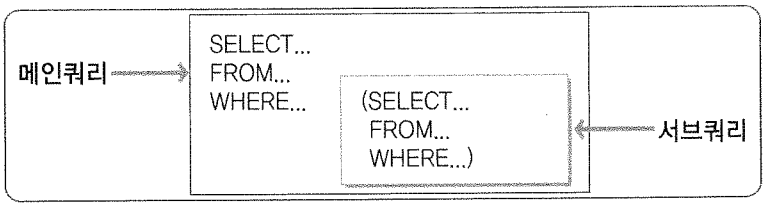
A	B	C	D
1	a	1	x
2	a	1	x
3	b	2	y

DEPT 테이블

D	E	F
x	i	5
y	m	6

① 0
② 1
③ 2
④ 3

메인쿼리와 서브쿼리



반환되는 데이터의 형태에 따른 서브쿼리 분류

서브쿼리 종류	설명
Single Row 서브쿼리 (단일 행 서브쿼리)	서브쿼리의 실행 결과가 항상 1건 이하인 서브쿼리를 의미한다. 단일 행 서브쿼리는 단일 행 비교 연산자와 함께 사용된다. 단일 행 비교 연산자에는 =, <, <=, >, >=, <>이 있다.
Multi Row 서브쿼리 (다중 행 서브쿼리)	서브쿼리의 실행 결과가 여러 건인 서브쿼리를 의미한다. 다중 행 서브쿼리는 다중 행 비교 연산자와 함께 사용된다. 다중 행 비교 연산자에는 IN, ALL, ANY, SOME, EXISTS가 있다.
Multi Column 서브쿼리 (다중 칼럼 서브쿼리)	서브쿼리의 실행 결과로 여러 칼럼을 반환한다. 메인쿼리의 조건절에 여러 칼럼을 동시에 비교할 수 있다. 서브쿼리와 메인쿼리에서 비교하고자 하는 칼럼 개수와 칼럼의 위치가 동일해야 한다.

95 아래는 서브쿼리에 대한 설명이다. 다음 중 올바른 것끼리 묶인 것은?

아 래

가) 서브쿼리는 단일 행(Single Row) 또는 복수 행(Multi Row) 비교 연산자와 함께 사용할 수 있다.
나) 서브쿼리는 SELECT 절, FROM 절, HAVING 절, ORDER BY 절 등에서 사용이 가능하다.
다) 서브쿼리의 결과가 복수 행(Multi Row) 결과를 반환하는 경우에는 '=', '<=', '=>' 등의 연산자와 함께 사용이 가능하다.
라) 연관(Correlated) 서브쿼리는 서브쿼리가 메인쿼리 컬럼을 포함하고 있는 형태의 서브쿼리이다.
마) 다중 컬럼 서브쿼리는 서브쿼리의 결과로 여러 개의 컬럼이 반환되어 메인쿼리의 조건과 동시에 비교되는 것을 의미하며 Oracle 및 SQL Server 등의 DBMS에서 사용 할 수 있다.

① 나, 라, 마
② 가, 나, 라
③ 나, 다, 라
④ 가, 나, 마

96 아래 테이블은 어느 회사의 사원들과 이들이 부양하는 가족에 대한 것으로 밑줄 친 칼럼은 기본키(Primary Key)를 표시한 것이다. 다음 중 '현재 부양하는 가족들이 없는 사원들의 이름을 구하라'는 질의에 대해 아래 SQL 문장의 ㉠, ㉡ 에 들어 갈 내용으로 가장 적절한 것은?

아 래

[테이블]
사원 (<u>사번</u>, 이름, 나이)
가족 (<u>이름</u>, 나이, 부양사번)
※ 가족 테이블의 부양사번은 사원 테이블의 사번을 참조하는 외래키(Foreign Key)이다.

[SQL 문장]
SELECT 이름
FROM 사원
WHERE ㉠ (SELECT * FROM 가족 WHERE ㉡)

① ㉠ : EXISTS ㉡ : 사번 = 부양사번
② ㉠ : EXISTS ㉡ : 사번 ◇ 부양사번
③ ㉠ : NOT EXISTS ㉡ : 사번 = 부양사번
④ ㉠ : NOT EXISTS ㉡ : 사번 ◇ 부양사번

핵심정리

서브쿼리를 사용시 주의사항

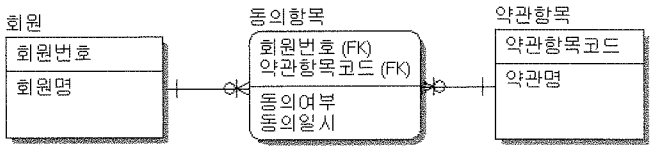
① 서브쿼리를 괄호로 감싸서 사용한다.

② 서브쿼리는 단일 행(Single Row) 또는 복수 행(Multiple Row) 비교 연산자와 함께 사용 가능하다. 단일 행 비교 연산자는 서브쿼리의 결과가 반드시 1건 이하이어야 하고 복수 행 비교 연산자는 서브쿼리의 결과 건수와 상관없다.

③ 서브쿼리에서는 ORDER BY를 사용하지 못한다. ORDER BY절은 SELECT절에서 오직 한 개만 올 수 있기 때문에 ORDER BY절은 메인쿼리의 마지막 문장에 위치해야 한다.

97 다음 중 아래의 ERD를 참조하여 아래 SQL과 동일한 결과를 출력하는 SQL로 가장 부적절한 것은?

아 래



[SQL]
```
SELECT A.회원번호, A.회원명
FROM   회원 A, 동의항목 B
WHERE  A.회원번호 = B.회원번호
GROUP BY A.회원번호, A.회원명
HAVING COUNT(CASE WHEN B.동의여부 = 'N' THEN 0 ELSE
       NULL END) >= 1
ORDER BY A.회원번호;
```

①
```
SELECT A.회원번호, A.회원명
FROM   회원 A
WHERE  EXISTS (SELECT 1 FROM 동의항목 B
                        WHERE  A.회원번호 = B.회원번호 AND
                               B.동의여부 = 'N')
ORDER BY A.회원번호;
```

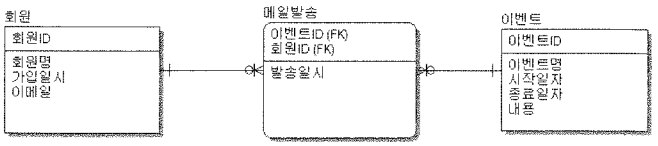
②
```
SELECT  A.회원번호, A.회원명
FROM    회원 A
WHERE   A.회원번호 IN (SELECT B.회원번호 FROM 동의항목 B
                       WHERE  B.동의여부 = 'N')
ORDER BY A.회원번호;
```

③
```
SELECT  A.회원번호, A.회원명
FROM    회원 A
WHERE   0 < (SELECT COUNT(*)
             FROM   동의항목 B WHERE B.동의여부 = 'N')
ORDER BY A.회원번호;
```

④
```
SELECT A.회원번호, A.회원명
FROM   회원 A, 동의항목 B
WHERE  A.회원번호 = B.회원번호 AND B.동의여부 = 'N'
GROUP BY A.회원번호, A.회원명
ORDER BY A.회원번호;
```

98 아래의 데이터 모델을 기준으로 SQL을 작성하였다. 다음 중 아래의 SQL에 대해 가장 바르게 설명한 것은?

아 래



[SQL]

```
SELECT A.회원ID, A.회원명, A.이메일
FROM   회원 A
㉠ WHERE  EXISTS (SELECT  'X'
                  FROM   이벤트 B, 메일발송 C
                  WHERE  B.시작일자 >= '2014.10.01'
                  AND    B.이벤트ID = C.이벤트ID
㉡                 AND    A.회원ID = C.회원ID
㉢                 HAVING COUNT(*) < (SELECT COUNT(*)
                                      FROM   이벤트
                                      WHERE  시작일자 >=
                                         '2014.10.01'));
```

① 이벤트 시작일자가 '2014.10.01'과 같거나 큰 이벤트를 대상으로 이메일이 발송된 기록이 있는 모든 회원을 추출하는 SQL이다.

② ㉡을 제거하고 ㉠의 EXISTS 연산자를 IN연산자로 변경해도 결과는 동일하다.

③ ㉢은 이벤트 시작일자가 '2014.10.01'과 같거나 큰 이벤트건수와 그 이벤트들을 기준으로 회원별 이메일 발송건수를 비교하는 것이다.

④ GROUP BY 및 집계함수를 사용하지 않고 HAVING절을 사용하였으므로 SQL이 실행되지 못하고 오류가 발생한다.

99 다음 중 서브쿼리에 대한 설명으로 가장 적절한 것은?

① 단일 행 서브쿼리는 서브쿼리의 실행 결과가 항상 한 건 이하인 서브쿼리로서 IN, ALL 등의 비교 연산자를 사용하여야 한다.

② 다중 행 서브쿼리 비교 연산자는 단일 행 서브쿼리의 비교 연산자로도 사용할 수 있다.

③ 연관 서브쿼리는 주로 메인쿼리에 값을 제공하기 위한 목적으로 사용한다.

④ 서브 쿼리는 항상 메인쿼리에서 읽혀진 데이터에 대해 서브쿼리에서 해당 조건이 만족하는지를 확인하는 방식으로 수행된다.

100 다음 중 아래 SQL에 대한 설명으로 가장 부적절한 것은?

아 래

```
[SQL]
SELECT B.사원번호, B.사원명, A.부서번호, A.부서명
      ,(SELECT COUNT(*) FROM 부양가족 Y WHERE
                Y.사원번호 = B.사원번호) AS 부양가족수
FROM  부서 A, (SELECT *
               FROM   사원
               WHERE 입사년도 = '2014') B
WHERE A.부서번호 = B.부서번호
AND   EXISTS (SELECT 1 FROM 사원 X WHERE X.부서번호 =
       A.부서번호);
```

① 위 SQL에는 다중 행 연관 서브쿼리, 단일 행 연관 서브쿼리, Inline View 가 사용되었다.

② SELECT절에 사용된 서브쿼리는 스칼라 서브쿼리라고도하며, 이러한 형태의 서브쿼리는 JOIN 으로 동일한 결과를 추출할 수도 있다.

③ WHERE 절의 서브쿼리에 사원 테이블 검색 조건으로 입사년도 조건을 FROM절의 서브쿼리와 동일하게 추가해야 원하는 결과를 추출할 수 있다.

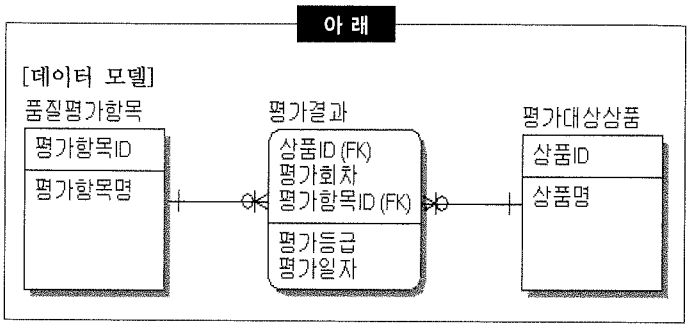
④ FROM 절의 서브쿼리는 동적 뷰(Dynamic View)라고도 하며, SQL 문장 중 테이블 명이 올 수 있는 곳에서 사용할 수 있다.

핵심정리

FROM 절에서 사용되는 서브쿼리를 인라인 뷰(Inline View)라고 한다. 서브쿼리의 결과가 마치 실행 시에 동적으로 생성된 테이블인 것처럼 사용할 수 있다. 인라인 뷰는 SQL문이 실행될 때만 임시적으로 생성되는 동적인 뷰이기 때문에 데이터베이스에 해당 정보가 저장되지 않는다.

서브쿼리의 결과가 마치 실행 시에 동적으로 생성된 테이블인 것처럼 사용할 수 있다. 인라인 뷰는 SQL문이 실행될 때만 임시적으로 생성되는 동적인 뷰이기 때문에 데이터베이스에 해당 정보가 저장되지 않는다.

101 **아래와 같은 데이터 모델에서 평가대상상품에 대한 품질평가항목별 최종 평가결과를 추출하는 SQL 문장으로 옳은 것은?(단, 평가항목에 대한 평가(평가등급)가 기대수준에 미치지 못할 경우 해당 평가항목에 대해서만 재평가를 수행한다)**



①
```
SELECT B.상품ID, B.상품명, C.평가항목ID, C.평가항목명, A.평가회차,
       A.평가등급, A.평가일자
FROM   평가결과 A, 평가대상상품 B, 품질평가항목 C,
       (SELECT MAX(평가회차) AS 평가회차 FROM 평가결과) D
WHERE A.상품ID = B.상품ID
AND    A.평가항목ID = C.평가항목ID
AND    A.평가회차 = D.평가회차;
```

②
```
SELECT B.상품ID, B.상품명, C.평가항목ID, C.평가항목명, A.평가회차,
       A.평가등급, A.평가일자
FROM   평가결과 A, 평가대상상품 B, 품질평가항목 C
WHERE A.상품ID = B.상품ID
AND    A.평가항목ID = C.평가항목ID
AND    A.평가회차 = (SELECT MAX(X.평가회차)
                     FROM   평가결과 X
                     WHERE X.상품ID = B.상품ID
                      AND    X.평가항목ID = C.평가항목ID);
```

③
```
SELECT B.상품ID, B.상품명, C.평가항목ID, C.평가항목명
      ,MAX(A.평가회차) AS 평가회차
            ,MAX(A.평가등급) AS 평가등급
      ,MAX(A.평가일자) AS 평가일자
FROM   평가결과 A, 평가대상상품 B, 품질평가항목 C
WHERE  A.상품ID = B.상품ID
AND    A.평가항목ID = C.평가항목ID
GROUP BY B.상품ID, B.상품명, C.평가항목ID, C.평가항목명;
```

④
```
SELECT B.상품ID, B.상품명, C.평가항목ID, C.평가항목명, A.평가회차,
       A.평가등급, A.평가일자
FROM  (SELECT 상품ID, 평가항목ID
            ,MAX(평가회차) AS 평가회차
                  ,MAX(평가등급) AS 평가등급
                  ,MAX(평가일자) AS 평가일자
       FROM   평가결과
       GROUP BY 상품ID, 평가항목ID) A, 평가대상상품 B, 품질평가항목 C
WHERE A.상품ID = B.상품ID
AND   A.평가항목ID = C.평가항목ID;
```

102 아래 부서 테이블의 담당자 변경을 위해 부서임시 테이블에 입력된 데이터를 활용하여 주기적으로 부서 테이블을 아래 결과와 같이 반영하기 위한 SQL으로 가장 적절한 것은?(단, 부서임시 테이블에서 변경일자를 기준으로 가장 최근에 변경된 데이터를 기준으로 부서 테이블에 반영되어야 한다)

아 래

[테이블 : 부서]

부서코드(PK)	부서명	상위부서코드	담당자
A001	대표이사	NULL	김대표
A002	영업본부	A001	홍길동
A003	경영지원본부	A001	이순신
A004	마케팅본부	A001	강감찬
A005	해외영업팀	A002	이청용
A006	국내영업팀	A002	박지성
A007	총무팀	A003	차두리
A008	인사팀	A003	이민정
A009	해외마케팅팀	A004	이병헌
A010	국내마케팅팀	A004	차승원

[테이블 : 부서임시]

변경일자(PK)	부서코드(PK)	담당자
2014.01.23	A007	이달자
2015.01.25	A007	홍경민
2015.01.25	A008	유재석

[결과]

부서코드(PK)	부서명	상위부서코드	담당자
A001	대표이사	NULL	김대표
A002	영업본부	A001	홍길동
A003	경영지원본부	A001	이순신
A004	마케팅본부	A001	강감찬
A005	해외영업팀	A002	이청용
A006	국내영업팀	A002	박지성
A007	총무팀	A003	홍경민
A008	인사팀	A003	유재석
A009	해외마케팅팀	A004	이병헌
A010	국내마케팅팀	A004	차승원

핵심정리

뷰 사용의 장점

- **독립성** : 테이블 구조가 변경되어도 뷰를 사용하는 응용 프로그램은 변경하지 않아도 된다.
- **편리성** : 복잡한 질의를 뷰로 생성함으로써 관련 질의를 단순하게 작성할 수 있다. 또한 해당 형태의 SQL문을 자주 사용할 때 뷰를 이용하면 편리하게 사용할 수 있다.
- **보안성** : 직원의 급여정보와 같이 숨기고 싶은 정보가 존재한다면, 뷰를 생성할 때 해당 칼럼을 빼고 생성함으로써 사용자에게 정보를 감출 수 있다.

```
① UPDATE 부서 A SET 담당자 = (SELECT C.부서코드
        FROM   (SELECT 부서코드, MAX(변경일자) AS 변경일자
                FROM   부서임시
                GROUP BY 부서코드) B, 부서임시 C
        WHERE  B.부서코드 = C.부서코드
        AND    B.변경일자 = C.변경일자
        AND    A.부서코드 = C.부서코드)
② UPDATE 부서 A SET 담당자 = (SELECT C.부서코드
        FROM   (SELECT 부서코드, MAX(변경일자) AS 변경일자
                FROM   부서임시
                GROUP BY 부서코드) B, 부서임시 C
        WHERE B.부서코드 = C.부서코드
        AND    B.변경일자 = C.변경일자
        AND    A.부서코드 = C.부서코드)
    WHERE  EXISTS (SELECT 1 FROM 부서 X WHERE A.부서코드 =
                   X.부서코드);
③ UPDATE 부서 A SET 담당자 = (SELECT B.담당자
        FROM   부서임시 B
        WHERE B.부서코드 = A.부서코드
        AND    B.변경일자 = (SELECT MAX(C.변경일자)
                             FROM 부서임시 C WHERE
                             C.부서코드 = B.부서코드))
    WHERE  부서코드 IN (SELECT 부서코드 FROM 부서임시);
④ UPDATE 부서 A SET 담당자 = (SELECT B.담당자
        FROM   부서임시 B
        WHERE B.부서코드 = A.부서코드
        AND    B.변경일자 = '2015.01.25.');
```

103 다음 중 뷰(View)에 대한 설명으로 가장 부적절한 것은?

① 뷰는 단지 정의만을 가지고 있으며, 실행 시점에 질의를 재작성하여 수행한다.
② 뷰는 복잡한 SQL 문장을 단순화 시켜주는 장점이 있는 반면, 테이블 구조가 변경되면 응용 프로그램을 변경해 주어야 한다.
③ 뷰는 보안을 강화하기 위한 목적으로도 활용할 수 있다.
④ 실제 데이터를 저장하고 있는 뷰를 생성하는 기능을 지원하는 DBMS도 있다.

104 아래 테이블에 대한 [뷰 생성 스크립트]를 실행한 후, 조회 SQL의 실행결과로 맞는 것은?

아 래

[TBL]

C1	C2
A	100
B	200
B	100
B	
	200

[뷰 생성 스크립트]

```
CREATE VIEW V_TBL
AS
SELECT *
FROM TBL
WHERE C1 = 'B' OR C1 IS NULL
```

[조회 SQL]

```
SELECT SUM(C2) C2
FROM V_TBL
WHERE C2 >= 200 AND C1 = 'B'
```

① 0 ② 200

③ 300 ④ 400

105 다음 중 아래의 테이블에서 SQL을 실행할 때 결과로 가장 적절한 것은?

아 래

[테이블 : 서비스]

서비스ID(PK)	서비스명
001	서비스1
002	서비스2
003	서비스3
004	서비스4

[테이블 : 서비스가입]

회원번호(PK)	서비스ID(PK)	가입일자
1	001	2013-01-01
1	002	2013-01-02
2	001	2013-01-01
2	002	2013-01-02
2	003	2013-01-03
3	001	2013-01-01
3	002	2013-01-02
3	003	2013-01-03

```
[SQL]
SELECT CASE WHEN GROUPING(A.서비스ID) = 0 THEN A.서비스ID
            ELSE '합계' END AS 서비스ID
      ,CASE WHEN GROUPING(B.가입일자) = 0
            THEN NVL(B.가입일자, '-') ELSE '소계' END AS 가입일자
      ,COUNT(B.회원번호) AS 가입건수
FROM  서비스 A LEFT OUTER JOIN 서비스가입 B
ON    (      A.서비스ID = B.서비스ID
        AND B.가입일자 BETWEEN '2013-01-01' AND '2013-01-31')
GROUP BY ROLLUP (A.서비스ID, B.가입일자);
```

①

서비스ID	가입일자	가입건수
001	2013-01-01	3
001	소계	3
002	2013-01-02	3
002	소계	3
003	2013-01-03	2
003	소계	2
합계	소계	8

②

서비스ID	가입일자	가입건수
001	2013-01-01	3
002	2013-01-02	3
003	2013-01-03	2
합계	소계	8

③

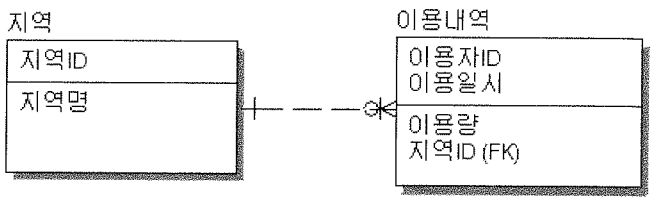
서비스ID	가입일자	가입건수
001	2013-01-01	3
001	소계	3
002	2013-01-02	3
002	소계	3
003	2013-01-03	2
003	소계	2
004	-	0
004	소계	0
합계	소계	8

④

서비스ID	가입일자	가입건수
001	2013-01-01	3
002	2013-01-02	3
003	2013-01-03	2
004	-	0
합계	소계	8

106 아래의 데이터 모델에서 SQL을 이용하여 표(지역별 월별 이용량)와 같은 형식의 데이터를 추출하려고 할 때 올바른 SQL 문장은?

아 래



[표 : 지역별 월별 이용량]

지역명	이용월	이용량
서울	2014.01	1,000
서울	2014.02	1,000
서울	월별합계	2,000
경기	2014.01	1,000
경기	2014.03	2,000
경기	월별합계	3,000
대전	2014.05	1,500
대전	2014.06	1,000
대전	월별합계	2,500
지역전체	월별합계	7,500

①
```
SELECT  (CASE GROUPING(B.지역명) WHEN 0 THEN '지역전체'
              ELSE B.지역명 END) AS 지역명
       ,(CASE GROUPING(TO_CHAR(A.이용일시, 'YYYY.MM'))
              WHEN 0 THEN '월별합계'
              ELSE TO_CHAR(A.이용일시, 'YYYY.MM') END) AS 이용월
       ,SUM(A.이용량) AS 이용량
FROM   이용내역 A INNER JOIN 지역 B ON (A.지역ID = B.지역ID)
GROUP BY ROLLUP(B.지역명, TO_CHAR(A.이용일시, 'YYYY.MM'))
```

②
```
SELECT  (CASE GROUPING(B.지역ID) WHEN 1 THEN '지역전체'
         ELSE MIN(B.지역명) END) AS 지역명
       ,(CASE GROUPING(TO_CHAR(A.이용일시, 'YYYY.MM'))
              WHEN 1 THEN '월별합계'
              ELSE TO_CHAR(A.이용일시, 'YYYY.MM') END) AS 이용월
       ,SUM(A.이용량) AS 이용량
FROM   이용내역 A INNER JOIN 지역 B ON (A.지역ID = B.지역ID)
GROUP BY ROLLUP(B.지역ID, TO_CHAR(A.이용일시, 'YYYY.MM'))
```

③
```
SELECT  (CASE GROUPING(B.지역명) WHEN 1 THEN '지역전체'
              ELSE B.지역명 END) AS 지역명
       ,(CASE GROUPING(TO_CHAR(A.이용일시, 'YYYY.MM'))
              WHEN 1 THEN '월별합계'
              ELSE TO_CHAR(A.이용일시, 'YYYY.MM') END) AS 이용월
       ,SUM(A.이용량) AS 이용량
FROM   이용내역 A INNER JOIN 지역 B ON (A.지역ID = B.지역ID)
GROUP BY CUBE(B.지역명, TO_CHAR(A.이용일시, 'YYYY.MM'))
```

④
```
SELECT  (CASE GROUPING(B.지역ID) WHEN 1 THEN '지역전체'
              ELSE MIN(B.지역명) END) AS 지역명
       ,(CASE GROUPING(TO_CHAR(A.이용일시, 'YYYY.MM'))
              WHEN 1 THEN '월별합계'
              ELSE TO_CHAR(A.이용일시, 'YYYY.MM') END) AS 이용월
       ,SUM(A.이용량) AS 이용량
FROM   이용내역 A INNER JOIN 지역 B ON (A.지역ID = B.지역ID)
GROUP BY GROUPING SETS(B.지역ID, TO_CHAR(A.이용일시, 'YYYY.MM'))
```

107 아래 결과를 얻기 위한 SQL문에서 ㉠ 에 들어갈 함수를 작성하시오.

아 래

구매이력

구매고객	구매월	총 구매건	총 구매액
AAA	201001	1	1000
AAA	201002	2	3000
AAA	201003	1	1000
AAA		4	5000
BBB	201001	3	2000
BBB	201002	5	3000
BBB	201003	1	2000
BBB		9	7000
CCC	201101	1	2000
CCC	201102	1	5000
CCC	201103	1	1000
CCC		3	8000
		16	20000

[SQL문]

```
SELECT 구매고객, 구매월, COUNT(*) "총 구매건", SUM(구매금액) "총 구매액"
FROM 구매이력
GROUP BY [ ㉠ ] (구매고객, 구매월)
```

핵심정리

Grouping Columns이 가질 수 있는 모든 경우에 대하여 Subtotal을 생성해야 하는 경우에는 CUBE를 사용하는 것이 바람직하나, ROLLUP에 비해 시스템에 많은 부담을 주므로 사용에 주의해야 한다.

CUBE는 결합 가능한 모든 값에 대하여 다차원 집계를 생성한다. CUBE도 결과에 대한 정렬이 필요한 경우는 ORDER BY 절에 명시적으로 정렬 칼럼이 표시가 되어야 한다.

108 다음 설명 중 가장 적절한 것은?

① 일반 그룹 함수를 사용하여 CUBE, GROUPING SETS와 같은 그룹 함수와 동일한 결과를 추출할 수 있으나, ROLLUP 그룹 함수와 동일한 결과는 추출할 수 없다.

② GROUPING SETS 함수의 경우에는 함수의 인자로 주어진 컬럼의 순서에 따라 결과가 달라지므로 컬럼의 순서가 중요하다.

③ CUBE, ROLLUP, GROUPING SETS 함수들의 대상 컬럼 중 집계된 컬럼 이외의 대상 컬럼 값은 해당 컬럼의 데이터 중 가장 작은 값을 반환한다.

④ CUBE 그룹 함수는 인자로 주어진 컬럼의 결합 가능한 모든 조합에 대해서 집계를 수행하므로 다른 그룹 함수에 비해 시스템에 대한 부하가 크다.

109 아래와 같이 설비와 에너지사용 테이블을 이용하여 결과를 나타내려할 때 SQL으로 가장 적절한 것을 2개 고르시오.

아 래

[테이블 : 설비]

설비ID	설비명
1	설비1
2	설비2
3	설비3

[테이블 : 에너지사용]

설비ID	에너지코드	사용량
1	전기	100
1	용수	200
1	바람	300
2	전기	200
2	용수	300
3	전기	300

[결과]

설비ID	에너지코드	사용량합계
1	바람	300
1	용수	200
1	전기	100
1	NULL	600
2	용수	300
2	전기	200
2	NULL	500
3	전기	300
3	NULL	300
NULL	바람	300
NULL	용수	500
NULL	전기	600
NULL	NULL	1400

① SELECT A.설비ID, B.에너지코드, SUM(B.사용량) AS 사용량합계
FROM 설비 A INNER JOIN 에너지사용량 B
ON (A.설비ID = B.설비ID)
GROUP BY CUBE ((A.설비ID), (B.에너지코드), (A.설비ID,
B.에너지코드))
ORDER BY A.설비ID, B.에너지코드;

② SELECT A.설비ID, B.에너지코드, SUM(B.사용량) AS 사용량합계
FROM 설비 A INNER JOIN 에너지사용량 B
ON (A.설비ID = B.설비ID)
GROUP BY CUBE (A.설비ID, B.에너지코드)
ORDER BY A.설비ID, B.에너지코드;

③ SELECT A.설비ID, B.에너지코드, SUM(B.사용량) AS 사용량합계
FROM 설비 A INNER JOIN 에너지사용량 B
ON (A.설비ID = B.설비ID)
GROUP BY GROUPING SETS((A.설비ID), (B.에너지코드),
(A.설비ID, B.에너지코드),())
ORDER BY A.설비ID, B.에너지코드;

④ SELECT A.설비ID, B.에너지코드, SUM(B.사용량) AS 사용량합계
FROM 설비 A INNER JOIN 에너지사용량 B
ON (A.설비ID = B.설비ID)
GROUP BY GROUPING SETS((A.설비ID), (B.에너지코드),
(A.설비ID, B.에너지코드))
ORDER BY A.설비ID, B.에너지코드;

핵심정리

GROUPING SETS은 다양한 소계 집합을 만들 수 있는데, GROUPING SETS에 표시된 인수들에 대한 개별 집계를 구할 수 있으며, 이때 표시된 인수들 간에는 계층 구조인 ROLLUP과는 달리 평등한 관계이므로 인수의 순서가 바뀌어도 결과는 같다.
그리고 GROUPING SETS 함수도 결과에 대한 정렬이 필요한 경우는 ORDER BY 절에 명시적으로 정렬 칼럼이 표시가 되어야 한다.

110 자재발주 테이블에 SQL을 수행하여 아래와 같은 결과를 얻었다. 다음 중 ㉠에 들어갈 문장으로 옳은 것은?

아 래

[테이블 : 자재발주]

자재번호	발주처ID	발주일자	발주수량
1	001	20150102	100
1	001	20150103	200
2	001	20150102	200
2	002	20150102	100
3	001	20150103	100
3	002	20150103	200

[SQL]

```
SELECT CASE WHEN GROUPING(자재번호) = 1 THEN '자재전체'
            ELSE 자재번호 END AS 자재번호
      ,CASE WHEN GROUPING(발주처ID) = 1 THEN '발주처전체'
            ELSE 발주처ID END AS 발주처ID
      ,CASE WHEN GROUPING(발주일자) = 1 THEN '발주일자전체'
            ELSE 발주일자 END AS 발주일자
      ,SUM(발주수량) AS 발주수량합계
FROM   자재발주
       ㉠
ORDER BY 자재번호, 발주처ID, 발주일자
```

[결과]

자재번호	발주처ID	발주일자	발주수량합계
1	발주처전체	발주일자전체	300
2	발주처전체	발주일자전체	300
3	발주처전체	발주일자전체	300
자재전체	001	20150102	300
자재전체	001	20150103	300
자재전체	002	20150102	100
자재전체	002	20150103	200

① GROUP BY CUBE (자재번호, (발주처ID, 발주일자))
② GROUP BY CUBE (자재번호, 발주처ID, 발주일자)
③ GROUP BY GROUPING SETS (자재번호, 발주처ID, 발주일자)
④ GROUP BY GROUPING SETS (자재번호, (발주처ID, 발주일자))

111 다음 중 월별매출 테이블을 대상으로 아래 SQL을 수행한 결과인 것은?

아 래

[테이블 : 월별매출]

상품ID	월	매출액
P001	2014.10	1500
P001	2014.11	1500
P001	2014.12	2500
P002	2014.10	1000
P002	2014.11	2000
P002	2014.12	1500
P003	2014.10	2000
P003	2014.11	1000
P003	2014.12	1000

[SQL]
```
SELECT 상품ID, 월, SUM(매출액) AS 매출액
FROM   월별매출
WHERE  월 BETWEEN '2014.10' AND '2014.12'
GROUP BY GROUPING SETS((상품ID, 월));
```

①

상품ID	월	매출액
NULL	2014.10	4500
NULL	2014.11	4500
NULL	2014.12	5000
P001	NULL	5500
P002	NULL	4500
P003	NULL	4000

②

상품ID	월	매출액
P001	2014.10	1500
P001	2014.11	1500
P001	2014.12	2500
P002	2014.10	1000
P002	2014.11	2000
P002	2014.12	1500
P003	2014.10	2000
P003	2014.11	1000
P003	2014.12	1000

③

상품ID	월	매출액
NULL	2014.10	4500
NULL	2014.11	4500
NULL	2014.12	5000
P001	NULL	5500
P002	NULL	4500
P003	NULL	4000
NULL	NULL	14000

④

상품ID	월	매출액
P001	2014.10	1500
P002	2014.10	1000
P003	2014.10	2000
NULL	2014.10	4500
P001	2014.11	1500
P002	2014.11	2000
P003	2014.11	1000
NULL	2014.11	4500
P001	2014.12	2500
P002	2014.12	1500
P003	2014.12	1000
NULL	2014.12	5000