

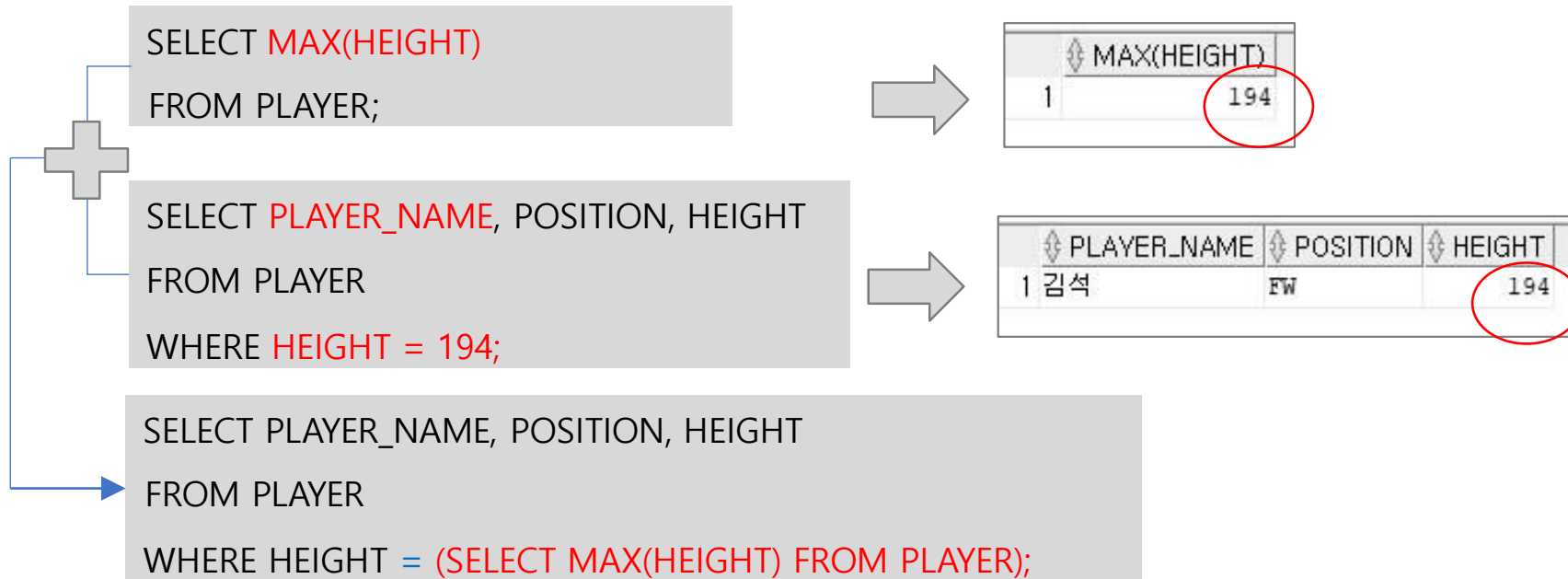
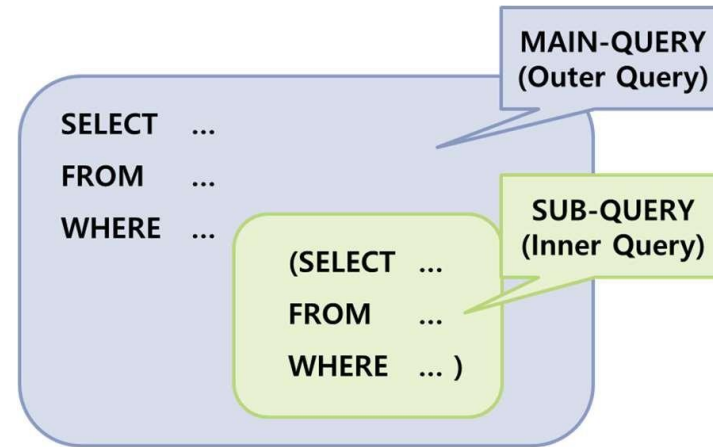
SQL – Subquery

문혜영

Subquery

- 서브쿼리

- SQL문 안에 포함된 SQL문
 - 예) 신장이 가장 큰 선수의 정보 조회



Subquery

- 서브쿼리

- SQL문 안에 포함된 SQL문

기준	유형
서브쿼리 위치	SELECT절, WHERE절 FROM절 (-> Inline View)
결과 칼럼 / 행의 수	단일행 서브쿼리 / 다중행 서브쿼리 단일칼럼 서브쿼리 / 다중칼럼 서브쿼리
메인 쿼리와의 연관성	연관(상관) 서브쿼리 / 비연관 서브쿼리

- 서브쿼리는 메인쿼리의 칼럼 모두 사용 가능
- 메인쿼리는 서브쿼리의 칼럼 사용 불가
 - Inline View에 정의된 칼럼만 사용 가능

- **결과 (칼럼/행)의 수에 따른 구분**

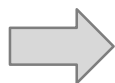
서브쿼리 종류	설명
Single Row (단일행)	서브쿼리의 실행 결과 로 항상 1건 이하 의 행을 반환 단일행 비교 연산자(= , < , <= , > , >= , <>)와 함께 사용
Multi Row (다중행)	서브쿼리의 실행 결과로 여러 건 의 행 반환 가능 다중행 비교 연산자(IN , ALL , ANY , SOME , EXISTS)와 함께 사용
Single Column (단일 칼럼)	서브쿼리의 실행 결과로 하나의 칼럼 을 반환
Multi Column (다중 칼럼)	서브쿼리의 실행 결과로 여러 칼럼 을 반환 서브쿼리와 메인쿼리의 비교 연산 수행 시, 비교하는 칼럼 개수와 위치가 동일해야 함

단일행 서브쿼리

● 단일행 서브쿼리

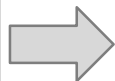
- 서브쿼리의 **결과** 건수가 반드시 **1건** 이하
- 단일행 비교 연산자(=, <, <=, >, >=, <>)와 함께 사용
 - 결과가 2건 이상이면 **Run Time** 오류 발생
- 예) '2007182'번 선수와 같은 팀에 속하는 선수의 이름, 포지션, 팀ID 출력

```
SELECT TEAM_ID  
FROM PLAYER  
WHERE PLAYER_ID = '2007182' ;
```



TEAM_ID
1 K06

```
SELECT PLAYER_NAME, TEAM_ID  
FROM PLAYER  
WHERE TEAM_ID =  
      (SELECT TEAM_ID  
       FROM PLAYER  
       WHERE PLAYER_ID = '2007182') ;
```



SQL 인출된 모든 행: 46(0초)		
PLAYER_NAME	POSITION	TEAM_ID
1 우르모브	DF	K06
2 윤희준	DF	K06
3 김규호	DF	K06
4 김민성	DF	K06
5 김장관	DF	K06
6 김정효	DF	K06
7 장대일	DF	K06

단일행 서브쿼리

- 단일행 서브쿼리

- Q) 사번 7499인 직원의 매니저를 찾아서

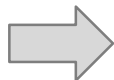
사번 7369인 직원의 매니저로 변경하는 질의를 완성하시오.

```
UPDATE EMP SET
```

```
???
```

```
WHERE EMPNO=7499;
```

	EMPNO	ENAME	MGR
1	7369	SMITH	7902
2	7499	ALLEN	7698
3	7521	WARD	7698
4	7566	JONES	7839
5	7654	MARTIN	7698
6	7698	BLAKE	7839
7	7782	CLARK	7839
8	7788	SCOTT	7566
9	7839	KING	(null)
10	7844	TURNER	7698
11	7876	ADAMS	7788
12	7900	JAMES	7698
13	7902	FORD	7566
14	7934	MILLER	7782



	EMPNO	ENAME	MGR
1	7369	SMITH	7902
2	7499	ALLEN	7902
3	7521	WARD	7698
4	7566	JONES	7839
5	7654	MARTIN	7698
6	7698	BLAKE	7839
7	7782	CLARK	7839
8	7788	SCOTT	7566
9	7839	KING	(null)
10	7844	TURNER	7698
11	7876	ADAMS	7788
12	7900	JAMES	7698
13	7902	FORD	7566
14	7934	MILLER	7782

SQL | 인출된 모든 행: 2(0.003초)

	EMPNO	ENAME	MGR
1	7369	SMITH	7902
2	7499	ALLEN	7902

단일행 서브쿼리

- 단일행 서브쿼리

- Q) 다음 질의에 대한 실행 결과를 예상하시오.

```
SELECT TEAM_ID  
FROM PLAYER  
WHERE PLAYER_NAME = '마니치' ;
```

???

```
SELECT TEAM_ID  
FROM PLAYER  
WHERE PLAYER_NAME = '김충호' ;
```

???

```
SELECT PLAYER_NAME, TEAM_ID  
FROM PLAYER  
WHERE TEAM_ID =  
      (SELECT TEAM_ID  
       FROM PLAYER  
       WHERE PLAYER_NAME = '마니치') ;
```

```
SELECT PLAYER_NAME, TEAM_ID  
FROM PLAYER  
WHERE TEAM_ID =  
      (SELECT TEAM_ID  
       FROM PLAYER  
       WHERE PLAYER_NAME = '김충호') ;
```

다중행 서브쿼리

● 다중행 서브쿼리

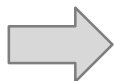
- 서브쿼리의 결과 건수가 **2건 이상일 가능성**이 있을 때
- 다중행 비교 연산자와 함께 사용
 - 2건 이상일 가능성은 있지만 결과 건수가 우연히 1개인 경우 → 단일행 비교 연산자도 에러는 발생하지 않음

다중행 연산자	설명
IN (서브쿼리)	임의의 결과 중 하나만 만족해도 참 (Multiple OR 조건)
비교연산자 ALL (서브쿼리)	결과의 모든 값 을 만족해야 하는 조건
비교연산자 ANY/SOME (서브쿼리)	결과의 어느 하나의 값이라도 만족하면 되는 조건 (ANY = SOME)
EXISTS (서브쿼리)	조건을 만족하는 값이 존재하는지 여부를 확인 조건을 만족하는 건을 하나라도 찾으면 검색 중지 (속도가 빠름)

다중행 서브쿼리

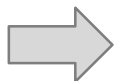
- IN 연산자

```
SELECT PLAYER_NAME, HEIGHT, BACK_NO  
FROM PLAYER  
WHERE BACK_NO = 15 ;
```



	PLAYER_NAME	HEIGHT	BACK_NO
1	윤희준	180	15
2	최철	176	15
3	정대수	184	15

```
SELECT PLAYER_NAME, HEIGHT, BACK_NO  
FROM PLAYER  
WHERE HEIGHT IN  
      (SELECT HEIGHT  
       FROM PLAYER  
       WHERE BACK_NO = 15) ;
```

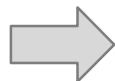


SQL 인출된 모든 행: 23(0,003초)			
	PLAYER_NAME	HEIGHT	BACK_NO
1	정진우	180	33
2	정광재	180	(null)
3	정남표	180	(null)
4	정동훈	180	(null)
5	정정수	180	36
6	정영근	180	5
7	정성근	180	33
8	남현우	180	31
9	장기봉	180	12
10	정태민	180	38
11	윤희준	180	15
12	우르모브	180	4
13	정학철	176	3
14	임영주	176	23

- ALL 연산자

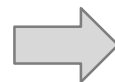
- 결과의 **모든** 값을 만족해야 하는 조건
 - 예) $x > \text{ALL}(1, 2, 3, 4, 5)$ 라면 $x > 5$ 가 되어야 함

```
SELECT PLAYER_NAME, HEIGHT, BACK_NO  
FROM PLAYER  
WHERE BACK_NO = 15 ;
```



PLAYER_NAME	HEIGHT	BACK_NO
1 윤희준	180	15
2 최철	176	15
3 정대수	184	15

```
SELECT PLAYER_NAME, HEIGHT, BACK_NO  
FROM PLAYER  
WHERE HEIGHT > ALL  
      (SELECT HEIGHT  
       FROM PLAYER  
       WHERE BACK_NO = 15) ;
```



PLAYER_NAME	HEIGHT	BACK_NO
1 김충호	185	(null)
2 김충호	185	60
3 정경진	186	41
4 정창오	186	27
5 박유석	186	1
6 정지혁	187	31
7 정재영	187	6
8 박상남	188	32
9 정용대	189	40
10 다오	190	61
11 우성용	191	22
12 이현	192	1
13 김석	194	20

다중행 서브쿼리

- ANY(=SOME) 연산자

- 결과의 어느 하나의 값이라도 만족하면 되는 조건 (ANY = SOME)
- 예) $x > \text{ANY}(1, 2, 3, 4, 5)$ 라면 $x > 1$ 이면 됨

- Q) 다음 질의로 출력된 결과 중 HEIGHT의 최소값은?

PLAYER_NAME	HEIGHT	BACK_NO
1 윤희준	180	15
2 최철	176	15
3 정대수	184	15

```
SELECT PLAYER_NAME, HEIGHT, BACK_NO
FROM PLAYER
WHERE HEIGHT >= ANY
( SELECT HEIGHT
  FROM PLAYER
  WHERE BACK_NO = 15 );
```

SQL 인출된 모든 행: 70(0.01초)			
	PLAYER_NAME	HEIGHT	BACK_NO
63	이영주	177	29
64	홍오균	177	22
65	김승희	177	5
66	이태성	177	30
67	최철	176	15
68	정상배	176	14
69	정학철	176	3
70	임영주	176	23

다중행 서브쿼리

● EXIST 연산자


- 조건을 만족하는 값이 **존재**하는지 여부를 **확인**
- 조건이 만족되는 1건만 찾으면 더 이상 검색하지 않음 (속도가 빠름)
- 주로 참/거짓의 조건 판단용으로 사용됨

```
SELECT PLAYER_NAME, HEIGHT, BACK_NO  
FROM PLAYER  
WHERE BACK_NO = 15 ;
```

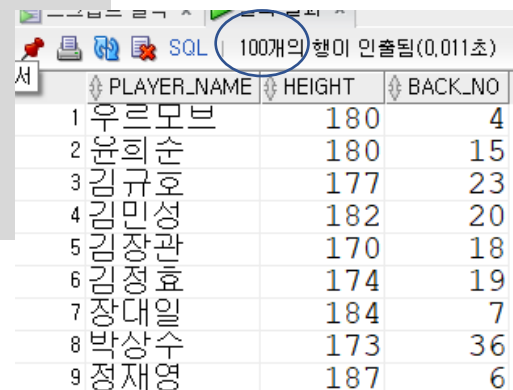


	PLAYER_NAME	HEIGHT	BACK_NO
1	윤희준	180	15
2	최철	176	15
3	정대수	184	15

```
SELECT PLAYER_NAME, HEIGHT, BACK_NO  
FROM PLAYER  
WHERE EXISTS (SELECT HEIGHT  
FROM PLAYER  
WHERE BACK_NO = 15) ;
```



```
SELECT PLAYER_NAME, HEIGHT, BACK_NO  
FROM PLAYER  
WHERE EXISTS (SELECT 1  
FROM PLAYER  
WHERE BACK_NO = 15) ;
```



	PLAYER_NAME	HEIGHT	BACK_NO
1	우르모브	180	4
2	윤희준	180	15
3	김규호	177	23
4	김민성	182	20
5	김장관	170	18
6	김정효	174	19
7	장대일	184	7
8	박상수	173	36
9	정재영	187	6

● 연관(Correlated) 서브쿼리

- 메인쿼리의 칼럼이 서브쿼리에서 사용된 쿼리
- Q) $AVG(SAL)$ 은 SAL 의 평균을 구하는 함수이다. 이 때 다음은 무엇을 조회하는 질의인가?

```
SELECT  ENAME, SAL, DEPTNO
FROM    EMP M
WHERE   SAL > (SELECT AVG(S.SAL)
               FROM EMP S
               WHERE M.DEPTNO = S.DEPTNO);
```

	ENAME	SAL	DEPTNO
1	ALLEN	1600	30
2	JONES	2975	20
3	BLAKE	2850	30
4	SCOTT	3000	20
5	KING	5000	10
6	FORD	3000	20

- Main에서 EMP M 을 Sub에 전달
- Sub에서 EMP M과 같은 부서인 EMP S의 평균 급여를 계산하여 Main에 전달
- Main에서 EMP M의 급여와 Sub에서 전달받은 급여를 비교

[참조]

```
SELECT  AVG(SAL)
FROM    EMP
WHERE   DEPTNO=30;
```

	AVG(SAL)
1	1566.6666

```
SELECT  SAL
FROM    EMP
WHERE   DEPTNO=30;
```

	SAL
1	1600
2	1250
3	1250
4	2850
5	1500
6	950

- 연관 서브쿼리의 특징

- 메인쿼리의 칼럼이 서브쿼리에서 사용된 쿼리
 - cf) 비연관 서브쿼리: 서브쿼리에서 메인쿼리의 칼럼을 사용하지 않음
- 메인쿼리가 먼저 수행되고, 그 후에 서브쿼리가 수행됨
 - 테이블의 별칭을 이용하여 메인 쿼리에서 서브쿼리로 정보 전달
 - 서브쿼리가 메인쿼리의 값을 이용, 그 후에 서브쿼리의 결과를 메인쿼리가 이용
- 서브쿼리에서 메인쿼리의 칼럼과 서브쿼리의 칼럼 간 비교가 이루어짐
 - 메인쿼리에서는 서브쿼리의 칼럼 사용 불가

```
SELECT  ENAME, SAL, DEPTNO
FROM    EMP M
WHERE   SAL > (SELECT AVG(S.SAL)
               FROM EMP S
               WHERE M.DEPTNO = S.DEPTNO);
```

다중컬럼 서브쿼리

● 다중컬럼 서브쿼리

- 서브쿼리의 결과로 여러 칼럼이 반환됨
- 예) PLAYER_ID가 2007188인 선수의 키, 포지션이 같은 선수 조회

```
SELECT    PLAYER_NAME, HEIGHT, POSITION, BACK_NO
FROM      PLAYER
WHERE     (HEIGHT, POSITION) =
          ( SELECT HEIGHT, POSITION
            FROM    PLAYER
            WHERE   PLAYER_ID = '2007188' );
```

	PLAYER_NAME	HEIGHT	POSITION	BACK_NO
1	우르모브	180	DF	4
2	윤희준	180	DF	15
3	정성근	180	DF	33
4	정영근	180	DF	5
5	정정수	180	DF	36
6	정동훈	180	DF	(null)
7	정남표	180	DF	(null)
8	정광재	180	DF	(null)
9	정진우	180	DF	33

[참조]

```
SELECT HEIGHT, POSITION
FROM    PLAYER
WHERE   PLAYER_ID = '2007188'
```

	HEIGHT	POSITION
1	180	DF

다중컬럼 다중행 서브쿼리

- 다중컬럼 다중행 서브쿼리

- 서브쿼리의 결과로 여러 컬럼이 반환될 때

```
SELECT    PLAYER_NAME, HEIGHT, POSITION
FROM      PLAYER
WHERE     (HEIGHT, POSITION) =
          (SELECT    HEIGHT, POSITION
           FROM      PLAYER
           WHERE     PLAYER_NAME = '마니치');
```

PLAYER_NAME	HEIGHT	POSITION
1 마니치	184	FW
2 정은중	184	FW

[참조]

```
SELECT    PLAYER_NAME, HEIGHT, POSITION
FROM      PLAYER
WHERE     PLAYER_NAME = '마니치';
```

PLAYER_NAME	HEIGHT	POSITION
1 마니치	184	FW

```
SELECT    PLAYER_NAME, HEIGHT, POSITION
FROM      PLAYER
WHERE     (HEIGHT, POSITION) =
          (SELECT    HEIGHT, POSITION
           FROM      PLAYER
           WHERE     PLAYER_NAME = '김충호');
```

ERROR!!!

[참조]

```
SELECT    PLAYER_NAME, HEIGHT, POSITION
FROM      PLAYER
WHERE     PLAYER_NAME = '김충호';
```

PLAYER_NAME	HEIGHT	POSITION
1 김충호	185	DF
2 김충호	185	GK

다중컬럼 다중행 서브쿼리

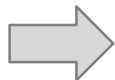
● 다중컬럼 다중행 서브쿼리

– Q) 부서별로 최고 급여를 받는 사원의 사원명, 부서번호, 급여를 출력하는 질의를 완성하시오.

- (단, 부서별 최고 급여를 구하는 질의는 아래 참고)

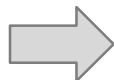
부서 번호 :DEPTNO, 급여:SAL

```
SELECT    DEPTNO, MAX(SAL)
FROM      EMP
GROUP BY  DEPTNO;
```



	DEPTNO	MAX(SAL)
1	30	2850
2	20	3000
3	10	5000

```
SELECT    ENAME, DEPTNO, SAL
FROM      EMP
WHERE     ???
```



	ENAME	DEPTNO	SAL
1	BLAKE	30	2850
2	FORD	20	3000
3	SCOTT	20	3000
4	KING	10	5000

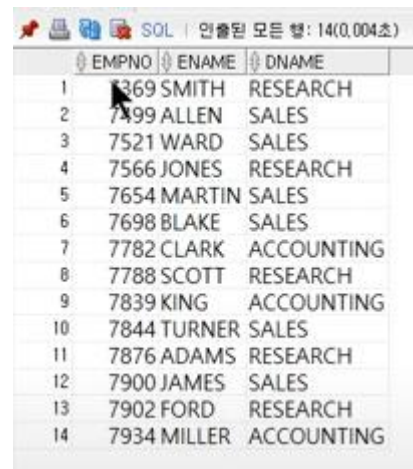
스칼라 서브쿼리

- 스칼라 서브쿼리(Scalar Subquery)

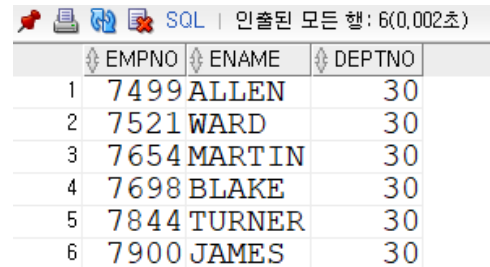
- 하나의 값을 반환하는 서브쿼리
 - 단일 행, 단일 칼럼
 - 하나의 값을 반환한다는 점에서 함수(Function)의 특성을 가짐
 - 공집합을 반환하는 경우 NULL이 대응됨
- 칼럼이 올 수 있는 대부분의 곳에서 사용 가능
 - SELECT절, WHERE절, 함수의 인자, ORDER BY절, CASE절, HAVING절 등
- 예) SELECT, WHERE 절에서 스칼라 서브쿼리의 이용

```
SELECT      EMPNO, ENAME,  
            (SELECT DNAME FROM DEPT WHERE DEPTNO = A.DEPTNO) AS DNAME  
FROM        EMP A;
```

```
SELECT      EMPNO, ENAME, DEPTNO  
FROM        EMP A  
WHERE       (SELECT DNAME FROM DEPT WHERE DEPTNO = A.DEPTNO) = 'SALES' ;
```



	EMPNO	ENAME	DNAME
1	7369	SMITH	RESEARCH
2	7499	ALLEN	SALES
3	7521	WARD	SALES
4	7566	JONES	RESEARCH
5	7654	MARTIN	SALES
6	7698	BLAKE	SALES
7	7782	CLARK	ACCOUNTING
8	7788	SCOTT	RESEARCH
9	7839	KING	ACCOUNTING
10	7844	TURNER	SALES
11	7876	ADAMS	RESEARCH
12	7900	JAMES	SALES
13	7902	FORD	RESEARCH
14	7934	MILLER	ACCOUNTING



	EMPNO	ENAME	DEPTNO
1	7499	ALLEN	30
2	7521	WARD	30
3	7654	MARTIN	30
4	7698	BLAKE	30
5	7844	TURNER	30
6	7900	JAMES	30

스칼라 서브쿼리

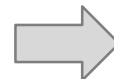
```
SELECT  EMPNO, ENAME,  
        (SELECT DNAME FROM DEPT WHERE DEPTNO = A.DEPTNO) AS DNAME  
FROM    EMP A;
```

DEPT

SELECT * FROM DEPT			
질의 결과 x			
SQL 인출된 모든 행: 4(0,003초)			
	DEPTNO	DNAME	LOC
1	10	ACCOUNTING	NEW YORK
2	20	RESEARCH	DALLAS
3	30	SALES	CHICAGO
4	40	OPERATIONS	BOSTON

EMP

질의 결과 x									
SQL 인출된 모든 행: 14(0,004초)									
	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	
1	7369	SMITH	CLERK	7902	80/12/17	800	(null)	20	
2	7499	ALLEN	SALESMAN	7698	81/02/20	1600	300	30	
3	7521	WARD	SALESMAN	7698	81/02/22	1250	500	30	
4	7566	JONES	MANAGER	7839	81/04/02	2975	(null)	20	
5	7654	MARTIN	SALESMAN	7698	81/09/28	1250	1400	30	
6	7698	BLAKE	MANAGER	7839	81/05/01	2850	(null)	30	
7	7782	CLARK	MANAGER	7839	81/06/09	2450	(null)	10	
8	7788	SCOTT	ANALYST	7566	87/07/13	3000	(null)	20	
9	7839	KING	PRESIDENT	(null)	81/11/17	5000	(null)	10	
10	7844	TURNER	SALESMAN	7698	81/09/08	1500	0	30	
11	7876	ADAMS	CLERK	7788	87/07/13	1100	(null)	20	
12	7900	JAMES	CLERK	7698	81/12/03	950	(null)	30	
13	7902	FORD	ANALYST	7566	81/12/03	3000	(null)	20	
14	7934	MILLER	CLERK	7782	82/01/23	1300	(null)	10	



	EMPNO	ENAME	DNAME
1	7369	SMITH	RESEARCH
2	7499	ALLEN	SALES
3	7521	WARD	SALES
4	7566	JONES	RESEARCH
5	7654	MARTIN	SALES
6	7698	BLAKE	SALES
7	7782	CLARK	ACCOUNTING
8	7788	SCOTT	RESEARCH
9	7839	KING	ACCOUNTING
10	7844	TURNER	SALES
11	7876	ADAMS	RESEARCH
12	7900	JAMES	SALES
13	7902	FORD	RESEARCH
14	7934	MILLER	ACCOUNTING

Company.sql

```
SELECT EMPNO, ENAME, DEPTNO
FROM EMP A
WHERE (SELECT DNAME FROM DEPT WHERE DEPTNO = A.DEPTNO) = 'SALES' ;
```

DEPT

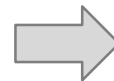
SQL | 인출된 모든 행: 4(0.003초)

	DEPTNO	DNAME	LOC
1	10	ACCOUNTING	NEW YORK
2	20	RESEARCH	DALLAS
3	30	SALES	CHICAGO
4	40	OPERATIONS	BOSTON

EMP

SQL | 인출된 모든 행: 14(0.004초)

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7369	SMITH	CLERK	7902	80/12/17	800	(null)	20
2	7499	ALLEN	SALESMAN	7698	81/02/20	1600	300	30
3	7521	WARD	SALESMAN	7698	81/02/22	1250	500	30
4	7566	JONES	MANAGER	7839	81/04/02	2975	(null)	20
5	7654	MARTIN	SALESMAN	7698	81/09/28	1250	1400	30
6	7698	BLAKE	MANAGER	7839	81/05/01	2850	(null)	30
7	7782	CLARK	MANAGER	7839	81/06/09	2450	(null)	10
8	7788	SCOTT	ANALYST	7566	87/07/13	3000	(null)	20
9	7839	KING	PRESIDENT	(null)	81/11/17	5000	(null)	10
10	7844	TURNER	SALESMAN	7698	81/09/08	1500	0	30
11	7876	ADAMS	CLERK	7788	87/07/13	1100	(null)	20
12	7900	JAMES	CLERK	7698	81/12/03	950	(null)	30
13	7902	FORD	ANALYST	7566	81/12/03	3000	(null)	20
14	7934	MILLER	CLERK	7782	82/01/23	1300	(null)	10



	EMPNO	ENAME	DEPTNO
1	7499	ALLEN	30
2	7521	WARD	30
3	7654	MARTIN	30
4	7698	BLAKE	30
5	7844	TURNER	30
6	7900	JAMES	30

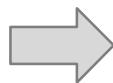
스칼라 서브쿼리

- 스칼라 서브쿼리(Scalar Subquery)

- 스칼라 서브쿼리는 함수의 인자로도 사용 가능
- Q) 앞 페이지의 상단 질의를 수정하여, 부서명을 3글자만 출력하는 질의를 작성하시오. (SUBSTR 함수 사용)

```
SELECT  EMPNO, ENAME,  
        ???  
FROM    EMP A;
```

	EMPNO	ENAME	D_NAME
1	7369	SMITH	RESEARCH
2	7499	ALLEN	SALES
3	7521	WARD	SALES
4	7566	JONES	RESEARCH
5	7654	MARTIN	SALES



	EMPNO	ENAME	D_NAME
1	7369	SMITH	RES
2	7499	ALLEN	SAL
3	7521	WARD	SAL
4	7566	JONES	RES
5	7654	MARTIN	SAL

뷰 (View)

● 뷰(View)

- 테이블은 실제로 데이터를 갖고 있지만, 뷰는 실제 데이터를 갖지 않음
 - 뷰 정의(View Definition, **SQL 텍스트 파일**)만 갖고 있음
 - 쿼리에서 뷰가 사용되면 DBMS 내부적으로 질의를 재작성(Rewrite)
- 실제 데이터를 가지고 있지 않지만 테이블의 역할 수행
 - 가상 테이블(Virtual Table)이라고도 함
- CREATE VIEW문을 통해 VIEW 생성

```
CREATE VIEW      V_PLAYER_TEAM AS
SELECT      P.PLAYER_NAME, P.BACK_NO, P.TEAM_ID, T.TEAM_NAME
FROM        PLAYER P INNER JOIN TEAM T
ON          P.TEAM_ID = T.TEAM_ID;
```

SQL | 50개의 행이 인출됨(0.004초)

	PLAYER_NAME	BACK_NO	TEAM_ID	TEAM_NAME
1	우르모브	4	K06	아이파크
2	윤희순	15	K06	아이파크
3	김규호	23	K06	아이파크
4	김민성	20	K06	아이파크
5	김장관	18	K06	아이파크
6	김정효	19	K06	아이파크

- VIEW의 확인

- View 결과 확인
- 뷰의 내용 확인
- 구조 확인

```
SELECT * FROM V_PLAYER_TEAM ;
```

```
SELECT * FROM USER_VIEWS;
```

```
DESC V_PLAYER_TEAM;
```

SQL | 인출된 모든 행: 1(0.13초)

VIEW_NAME	TEXT_LENGTH	TEXT
V_PLAYER_TEAM	114	SE

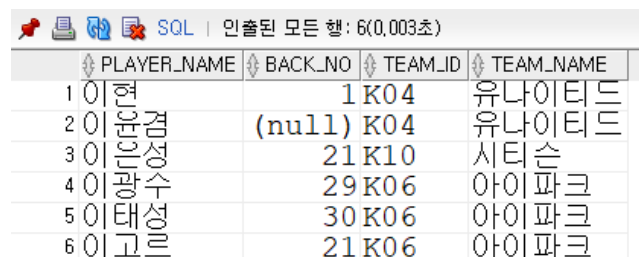
이름	널?	유형
PLAYER_NAME	NOT NULL	VARCHAR2(20)
BACK_NO		NUMBER(2)
TEAM_ID	NOT NULL	CHAR(3)
TEAM_NAME	NOT NULL	VARCHAR2(40)

뷰 (View)

● 뷰(View)

- 생성된 뷰는 테이블과 동일한 형태로 사용 가능

```
SELECT    PLAYER_NAME, BACK_NO, TEAM_ID, TEAM_NAME
FROM      V_PLAYER_TEAM
WHERE     PLAYER_NAME LIKE '이%';
```

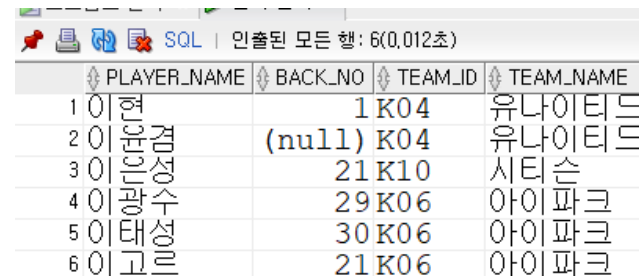


SQL | 인출된 모든 행: 6(0.003초)

	PLAYER_NAME	BACK_NO	TEAM_ID	TEAM_NAME
1	이영준	1	K04	유나이티드
2	이영준	(null)	K04	유나이티드
3	이광수	21	K10	시티즌
4	이태성	29	K06	아이파크
5	이태성	30	K06	아이파크
6	이고르	21	K06	아이파크

- 파싱 시점에 DBMS가 내부적으로 뷰 해당 부분을 SQL문으로 재작성

```
SELECT    PLAYER_NAME, BACK_NO, TEAM_ID, TEAM_NAME
FROM(     SELECT P.PLAYER_NAME, P.BACK_NO, P.TEAM_ID, T.TEAM_NAME
          FROM PLAYER P INNER JOIN TEAM T
          ON P.TEAM_ID = T.TEAM_ID )
WHERE     PLAYER_NAME LIKE '이%';
```



SQL | 인출된 모든 행: 6(0.012초)

	PLAYER_NAME	BACK_NO	TEAM_ID	TEAM_NAME
1	이영준	1	K04	유나이티드
2	이영준	(null)	K04	유나이티드
3	이광수	21	K10	시티즌
4	이광수	29	K06	아이파크
5	이태성	30	K06	아이파크
6	이고르	21	K06	아이파크

- VIEW의 제거

```
DROP VIEW V_PLAYER_TEAM;
```

뷰 (View)

- 뷰(View)

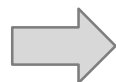
- 계층적 뷰 생성

- Q) 사원과 부서 테이블로부터 사원번호, 사원명, 부서번호, 부서명을 추출한 뷰 V_EMP_DEPT를 작성하시오.
또한 이 뷰로부터 사원명과 부서명 만을 다시 추출한 V_EMP_DEPT2를 작성하시오

SELECT * FROM V_EMP_DEPT;

SQL | 인출된 모든 행: 14(0,003초)

	EMPNO	ENAME	DEPTNO	DNAME
1	7369	SMITH	20	RESEARCH
2	7499	ALLEN	30	SALES
3	7521	WARD	30	SALES
4	7566	JONES	20	RESEARCH
5	7654	MARTIN	30	SALES
6	7698	BLAKE	30	SALES
7	7782	CLARK	10	ACCOUNTING
8	7788	SCOTT	20	RESEARCH
9	7839	KING	10	ACCOUNTING
10	7844	TURNER	30	SALES



SELECT * FROM V_EMP_DEPT2;

SQL | 인출된 모든 행: 14(0,003초)

	ENAME	DNAME
1	SMITH	RESEARCH
2	ALLEN	SALES
3	WARD	SALES
4	JONES	RESEARCH
5	MARTIN	SALES
6	BLAKE	SALES
7	CLARK	ACCOUNTING
8	SCOTT	RESEARCH
9	KING	ACCOUNTING
10	TURNER	SALES

사원번호: EMPNO,
사원명 : ENAME,
부서번호: DEPTNO,
부서명 : DNAME ,
사원테이블 : EMP
부서테이블 : DEPT

뷰 (View)

- 뷰(View)

뷰의 장점	설명
독립성	테이블 구조가 변경시, 뷰만 변경되고 뷰를 사용하는 응용 프로그램은 변경될 필요가 없음 (예: 앞에서 PLAYER와 TEAM 테이블이 변경된 경우 → V_PLAYER_TEAM 뷰가 이에 맞게 변경된다면 응용프로그램은 변경될 필요 없음)
편리성	복잡한 질의를 뷰로 생성하여 질의의 가독성을 높임
보안성	민감한 정보(급여정보 등)를 제외하고 뷰를 생성하여, 사용자로부터 정보를 보호할 수 있음

인라인 뷰 (Inline View)

- 인라인 뷰(Inline View)

- FROM 절에서 사용되는 서브쿼리
- 실행 순간에만 임시적으로 생성되며 DB에 저장되지 않음
 - 인라인 뷰(Inline View) = 동적 뷰(Dynamic View)
 - 일반 뷰 = 정적 뷰(Static View)
- 인라인 뷰의 SELECT문에서 정의된 칼럼은 메인 쿼리에서 사용 가능
 - cf) 일반적으로 서브쿼리에서 정의된 칼럼은 메인 쿼리에서 사용 불가

```
SELECT EMPNO  
FROM ( SELECT EMPNO, ENAME  
        FROM EMP  
        ORDER BY MGR);
```

OK!!

```
SELECT MGR  
FROM ( SELECT EMPNO, ENAME  
        FROM EMP  
        ORDER BY MGR);
```

ERROR!!

인라인 뷰 (Inline View)

- 인라인 뷰(Inline View)

- Q) 급여가 2,000 초과인 직원들에 대해 직원번호, 직원명, 급여, 부서명을 출력하고자 한다.

다음의 질의에서 오류를 수정하시오.

```
SELECT    E.EMPNO, E.ENAME, E.SAL, D.DNAME  
FROM      ( SELECT EMPNO, ENAME, SAL  
            FROM EMP  
            WHERE SAL > 2000 ) E, DEPT D  
WHERE     E.DEPTNO = D.DEPTNO;
```

ERROR!!



???

수고하셨습니다 🙌