

# SQL #2 - SQL Query II

---

Join (Inner, Outer, Natural)  
Group & Aggregation  
Subquery

JOIN

# Join


- 둘 이상의 테이블을 합쳐서 하나의 큰 테이블로 만드는 방법
- 필요성
  - 관계형 모델에서는 데이터의 일관성이나 효율을 위하여 데이터의 중복을 최소화 (정규화)
  - Foreign Key를 이용하여 참조
  - 정규화된 테이블로부터 결합된 형태의 정보를 추출할 필요가 있음
  - 예) 직원의 이름과 직원이 속한 부서명을 함께 보고 싶으면???

EMPNO	ENAME	DEPTNO
7369	SMITH	20
7499	ALLEN	30
7521	WARD	30
7566	JONES	20
7654	MARTIN	30
7698	BLAKE	30
7782	CLARK	10
7788	SCOTT	20
7839	KING	10
7844	TURNER	30
7876	ADAMS	20
7900	JAMES	30
7902	FORD	20
7934	MILLER	10

?

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

# 카티전 프로덕트

- 두 테이블에서 그냥 결과를 선택하면? 
  - SELECT ename, dname FROM emp, dept
  - 결과: 두 테이블의 행들의 가능한 모든 쌍이 추출됨
  - 일반적으로 사용자가 원하는 결과가 아님.
- Cartesian Product
$$X \times Y = \{(x, y) | x \in X \text{ and } y \in Y\}$$
- Cartesian Product를 막기 위해서는 올바른 Join조건을 WHERE 절에 부여 해야 함.

ENAME	DNAME
-----	-----
SMITH	ACCOUNTING
ALLEN	ACCOUNTING
WARD	ACCOUNTING
JONES	ACCOUNTING
MARTIN	ACCOUNTING
BLAKE	ACCOUNTING
CLARK	ACCOUNTING
SCOTT	ACCOUNTING
...	
ALLEN	OPERATIONS
WARD	OPERATIONS
JONES	OPERATIONS
MARTIN	OPERATIONS
BLAKE	OPERATIONS
CLARK	OPERATIONS
SCOTT	OPERATIONS
KING	OPERATIONS
TURNER	OPERATIONS
ADAMS	OPERATIONS
JAMES	OPERATIONS
FORD	OPERATIONS
MILLER	OPERATIONS

56 개의 행이 선택되었습니다.

# Simple Join

- Syntax

```
SELECT t1.col1, t1.col2, t2.col1 ...  
FROM Table1 t1, Table2 t2  
WHERE t1.col3 = t2.col3
```

- 설명

- FROM 절에 필요로 하는 테이블을 모두 적는다.
- 컬럼 이름의 모호성을 피하기 위해(어느 테이블에 속하는지 알 수 없음)이 있을 수 있으므로 Table 이름에 Alias 사용 (테이블 이름으로 직접 지칭 가능)
- 적절한 Join 조건을 Where 절에 부여 (일반적으로 테이블 개수 -1 개의 조인 조건이 필요)
- 일반적으로 PK와 FK간의 = 조건이 붙는 경우가 많음

# Join 처리 방법

Where절의 조인 조건을 이용  
From절의 테이블들을 Join하여  
임시 테이블을 만든다..

**FROM**

테이블로부터 한 Row를 읽는다.

row가 Where절의 조건을  
만족하는가?

FALSE

**WHERE**

TRUE

임시 결과 생성

ORDER BY 절을 이용 정렬

**ORDER BY**

SELECT절을 이용하여  
Projection

**SELECT**

테이블의 모든 Row를  
처리할 때까지 반복

실제 모든 SQL이 이렇게 처리되는 것은 아닙니다. SQL의 처리 순서는 DBMS가 질의 최적화 과정을 통하여 결정합니다. 질의의 종류, 데이터의 분포 등에 따라 질의의 실제 순서는 달라질 수도 있습니다.

# Join 종류

---

- 용어
  - Cross Join (Cartesian Product): 모든 가능한 쌍이 나타남
  - Inner Join: Join 조건을 만족하는 튜플만 나타남
  - Outer Join: Join 조건을 만족하지 않는 튜플 (짜이 없는 튜플)도 null과 함께 나타남
  - Theta Join: 조건(theta)에 의한 조인
  - Equi-Join: Theta Join & 조건이 Equal (=)
  - Natural Join: Equi-join & 동일한 Column명 합쳐짐.
  - Self Join: 자기 자신과 조인

# Equi-Join

EMPNO	ENAME	.....	DEPTNO
7839	KING		10
7566	JONES		20
7900	JAMES		30
7369	SMITH		20
7499	ALLEN		30

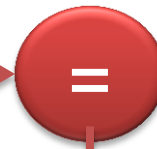
**EMP**

**FK**

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATION	BOSTON

**PK**

**DEPT**



EMPNO	ENAME	.....	DEPTNO	DEPTNO	DNAME	LOC
7839	KING		10	10	ACCOUNTING	NEW YORK
7566	JONES		20	20	RESEARCH	DALLAS
7900	JAMES		30	30	SALES	CHICAGO
7369	SMITH		20	20	RESEARCH	DALLAS
7499	ALLEN		30	30	SALES	CHICAGO

**SELECT \* FROM EMP, DEPT  
WHERE EMP.DEPTNO = DEPT.DEPTNO**



# Theta Join

- 정의
  - 임의의 조건을 Join 조건으로 사용가능
  - Non-Equi Join이라고도 함

예

```
SELECT e.ename, e.sal, s.grade
FROM emp e, salgrade s
WHERE e.sal BETWEEN s.losal AND s.hisal
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	ENAME	SAL	GRADE
7369	SMITH	CLERK	7902	80/12/17	800	SMITH	800	1
7499	ALLEN	SALESMAN	7698	81/02/20	1600	JAMES	950	1
7521	WARD	SALESMAN	7698	81/02/22	1250	ADAMS	1100	1
7566	JONES	MANAGER	7839	81/04/02	2975	WARD	1250	2
7654	MARTIN	SALESMAN	7698	81/09/28	1250	MARTIN	1250	2
7698	BLAKE	MANAGER	7839	81/05/01	2850	MILLER	1300	2
7782	CLARK	MANAGER	7839	81/06/09	2450	TURNER	1500	3
7788	SCOTT	ANALYST	7566	87/04/19	3000	ALLEN	1600	3
7839	KING	PRESIDENT		81/11/17	5000	CLARK	2450	4
7844	TURNER	SALESMAN	7698	81/09/08	1500	BLAKE	2850	4
7876	ADAMS	CLERK	7788	87/05/23	1100	JONES	2975	4
7900	JAMES	CLERK	7698	81/12/03	950	SCOTT	3000	4
7902	FORD	ANALYST	7566	81/12/03	3000	FORD	3000	4
7934	MILLER	CLERK	7782	82/01/23	1300	KING	5000	5

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

# Outer Join

---

- 정의
  - Join 조건을 만족하지 않는 (짜이 없는 ) 튜플의 경우 Null을 포함하여 결과를 생성
  - 모든 행이 결과 테이블에 참여
- 종류
  - Left Outer Join: 왼쪽의 모든 튜플은 결과 테이블에 나타남
  - Right Outer Join: 오른쪽의 모든 튜플은 결과 테이블에 나타남
  - Full Outer Join: 양쪽 모두 결과 테이블에 참여
- 표현 방법
  - NULL이 올 수 있는 쪽 조건에 (+)를 붙인다. (오라클)

# Outer Join

EMP			FK	PK	DEPT		
EMPNO	ENAME	.....	DEPTNO		DEPTNO	DNAME	LOC
7839	KING		10	→	10	ACCOUNTING	NEW YORK
7566	JONES		20	→	20	RESEARCH	DALLAS
7900	JAMES		30	→	30	SALES	CHICAGO
7369	SMITH		20	→	40	OPERATION	BOSTON

EMPNO	ENAME	SELECT * FROM EMP, DEPT					
7839	KING	WHERE EMP.DEPTNO (+) = DEPT.DEPTNO					
7566	JONES		20	20	RESEARCH	DALLAS	
7900	JAMES		30	30	SALES	CHICAGO	
7369	SMITH		20	20	RESEARCH	DALLAS	
7499	ALLEN		30	30	SALES	CHICAGO	
				40	OPERATION	BOSTON	

# Self Join

- 자기 자신과 Join
- Alias를 사용할 수 밖에 없음

```
SELECT * FROM EMP E1, EMP E2
WHERE E1.EMPNO = E2.MGR
```

PK	EMP	FK	
EMPNO	ENAME	MGR	.....
7839	KING		
7566	JONES	7839	
7900	JAMES	7698	
7369	SMITH	7902	
7499	ALLEN	7698	

EMPNO	ENAME	MGR	.....	EMPNO	ENAME
7566	JONES	7839		7839	KING
7900	JAMES	7698		7698	BLAKE
7369	SMITH	7902		7902	FORD
7499	ALLEN	7698		7698	BLAKE

# SQL:1999 Syntax (Oracle 9i)

- From절에서 바로 Join을 명시적으로 정의

```
SELECT table1.column, table2.column
FROM table1
[CROSS JOIN table2] |
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
ON(table1.column_name = table2.column_name)] |
[LEFT|RIGHT|FULL OUTER JOIN table2
ON (table1.column_name = table2.column_name)];
```

- 예
  - SELECT \* FROM emp **NATURAL JOIN** dept;
  - SELECT \* FROM emp **JOIN** dept **USING** (deptno);
  - SELECT \* FROM emp **JOIN** dept **ON** emp.deptno = dept.deptno;
  - SELECT \* FROM emp **RIGHT OUTER JOIN** dept **ON** (emp.deptno = dept.deptno);

# GROUP & AGGREGATION

# Aggregate Function (집계함수)

---

- 여러행으로부터 하나의 결과값을 반환
- 종류
  - AVG
  - COUNT
    - COUNT(\*): number of rows in table (NULL도 count된다)
    - COUNT(expr): non-null value (NULL은 빠진다)
    - COUNT(DISTINCT expr): distinct non-null
  - MAX
  - MIN
  - SUM
  - STDDEV
  - VARIANCE

# Aggregate Function

```
SELECT sal FROM emp;
```

SAL
800
1600
1250
2975
1250
2850
2450
3000
5000
1500
1100
950
3000
1300

```
SELECT AVG(sal) FROM emp;
```

AUG(SAL)
2073.21429



# 일반적인 오류

- `SELECT deptno, AVG(sal) FROM emp;`
- 주의
  - 집계함수의 결과는 한 row만 남게 된다.
  - deptno는 하나의 row에 표현될 수 없다.
  - 부서별과 같은 내용이 필요할 때는 **Group by**절 사용

# GROUP BY

```
SELECT deptno, sal  
FROM emp  
ORDER BY deptno;
```

DEPTNO	SAL
10	2450
10	5000
10	1300
20	2975
20	3000
20	1100
20	800
20	3000
30	1250
30	1500
30	1600
30	950
30	2850
30	1250

```
SELECT deptno, AVG(sal)  
FROM emp  
GROUP BY deptno  
ORDER BY deptno;
```

DEPTNO	AVG(SAL)
10	2916.66667
20	2175
30	1566.66667

# 일반적인 오류

- 부서별 월급에서 부서명도 출력?

```
SELECT deptno, dname, AVG(sal)
FROM emp
GROUP BY deptno
ORDER BY deptno;
```

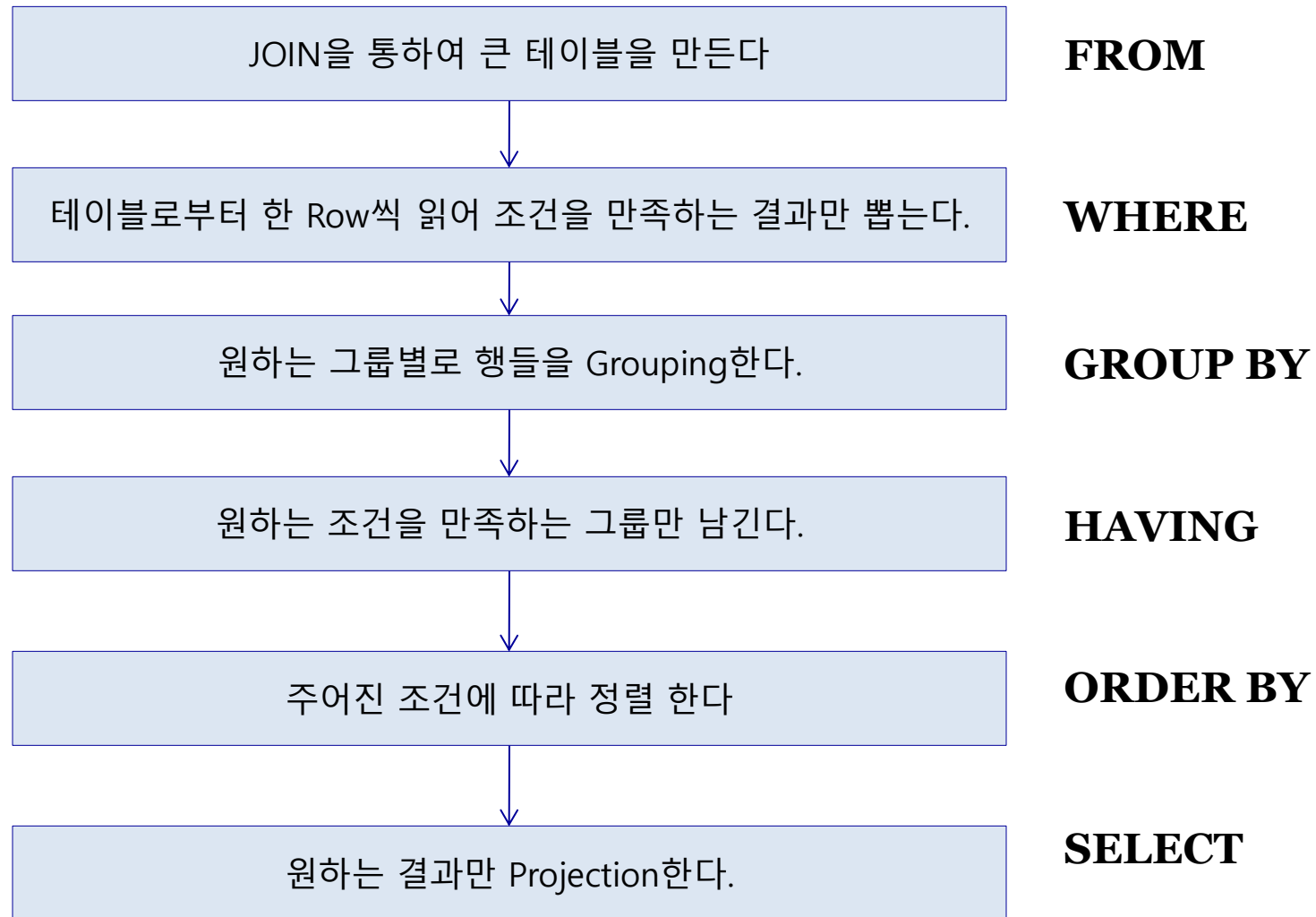
- 비록 부서번호에 따라 부서명은 하나로 결정될 수 있지만, dname은 grouping에 참여하지 않았으므로 하나의 row로 aggregate될 수 있다고 볼 수 없음
- 주의
  - SELECT 의 Col 리스트에는 Group by에 참여한 필드나 aggregate 함수만 올 수 있다.
  - Group by 이후에는 Group by에 참여한 필드나 aggregate 함수만 남아있는 셈
    - HAVING, ORDER BY 도 마찬가지

# HAVING 절

- Aggregation 결과에 대해 다시 condition을 검사할 때
- 일반적인 오류
  - 평균 월급이 2000 이상인 부서는?
  -
- 주의
  - WHERE 절은 Aggregation 이전, HAVING 절은 Aggregation 이후의 filtering
  - Having절에는 Group by에 참여한 컬럼이나 Aggregate 함수만 사용가능

```
SELECT deptno, AVG(sal)
FROM emp
WHERE AVG(sal) > 2000
GROUP BY deptno;
```

# 단일 SQL 문 실행 순서



# 단일 SQL 작성법

---

- ① 최종 출력될 정보에 따라 원하는 컬럼 SELECT 절에 추가
- ② 원하는 정보를 가진 테이블들을 FROM 절에 추가
- ③ WHERE절에 알맞은 Join 조건 추가
- ④ WHERE절에 알맞은 검색 조건 추가
- ⑤ 필요에 따라 GROUP BY, HAVING 등을 통해 Grouping
- ⑥ 정렬 조건 ORDER BY에 추가

# SUBQUERY

# Subquery

- 하나의 SQL 질의문 속에 다른 SQL 질의문이 포함되어 있는 형태
- 예) 'SCOTT'보다 월급이 많은 사람의 이름은?
  - 월급이 많은 사람의 이름?
    - SELECT ename FROM emp WHERE sal > ???
  - 'SCOTT'의 월급?
    - SELECT sal FROM emp WHERE ename='SCOTT'

```
SELECT ename
FROM emp
WHERE sal > ( SELECT sal
              FROM emp
              WHERE ename = 'SCOTT' )
```



# Single-Row Subquery

- Subquery의 결과가 한 ROW인 경우
- Single-Row Operator 사용해야 함: = , > , >=, < , <=, <>

```
SELECT  ename, sal, deptno
FROM    emp
WHERE   ename = (SELECT  MIN(ename) FROM emp);
```

```
SELECT  ename, sal
FROM    emp
WHERE   sal < (SELECT  AVG(sal) FROM emp);
```

```
SELECT  ename, deptno
FROM    emp
WHERE   deptno = (SELECT  deptno
                  FROM    dept
                  WHERE   dname = 'SALES');
```

# Multi-Row Query

- Subquery의 결과가 둘 이상의 Row
- Multi-Row에 대한 연산을 사용해야 함: ANY, ALL, IN, EXIST...

```
SELECT  ename, sal, deptno
FROM    emp
WHERE   ename = (SELECT  MIN(ename)
                  FROM    emp GROUP BY deptno) ;
```



```
SELECT  ename, sal, deptno
FROM    emp
WHERE   ename IN (SELECT  MIN(ename)
                  FROM    emp GROUP BY deptno) ;
```



```
SELECT  ename, sal, deptno
FROM    emp
WHERE   ename = ANY (SELECT  MIN(ename)
                     FROM    emp GROUP BY deptno) ;
```



# Correlated Query

- Outer Query와 Inner Query가 서로 연관되어 있음
- 해석방법
  - Outer query의 한 Row를 얻는다.
  - 해당Row를 가지고 Inner Query를 계산한다.
  - 계산 결과를 이용 Outer query의 WHERE절을 evaluate
  - 결과가 참이면 해당 Row를 결과에 포함시킨다.

```
SELECT  ename, sal, deptno
FROM    emp outer
WHERE   sal > (SELECT AVG(sal)
              FROM    emp
              WHERE   deptno = outer.deptno) ;
```

# 예제

- 각 부서별로 최고급여를 받는 사원을 출력하시오.

```
SELECT deptno, empno, ename, sal
FROM emp
WHERE (deptno,sal) IN (SELECT deptno, max(sal)
                      FROM emp GROUP BY deptno);
```

```
SELECT e.deptno, e.empno, e.ename, e.sal
FROM emp e, (SELECT s.deptno, max(s.sal)  msal
              FROM emp s GROUP BY deptno) m
WHERE e.deptno = m.deptno AND e.sal = m.msal;
```

```
SELECT deptno, empno, ename, sal
FROM emp e
WHERE e.sal = (SELECT max(sal)
              FROM emp WHERE deptno = e.deptno);
```

# Top-K Query (ORACLE)

- ROWNUM: 질의의 결과에 가상으로 부여되는 Oracle의 Pseudo Column
- Top-K Query: 조건을 만족하는 상위 k개의 결과를 빨리 얻기
  - 81년도에 입사한 사람 중 월급이 가장 많은 3명은 누구인가?

```
SELECT rownum, ename, sal
FROM emp
WHERE hiredate like '81%' AND rownum < 4
ORDER BY sal DESC;
```

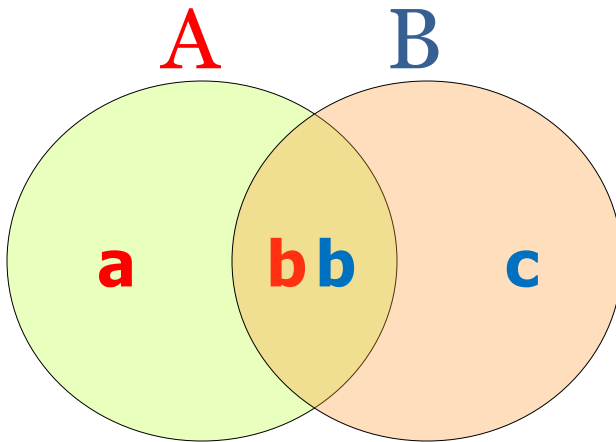


```
SELECT rownum, ename, sal
FROM (SELECT *
      FROM emp
      WHERE hiredate like '81%'
      ORDER BY sal DESC)
WHERE rownum < 4;
```



# SET Operator

- 두 질의의 결과를 가지고 집합 연산
- UNION, UNION ALL, INTERSECT, MINUS

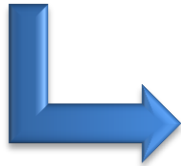


- $A \cup B = \{a, b, c\}$
- $A \cup ALL B = \{a, b, b, c\}$
- $A \cap B = \{b\}$
- $A - B = \{a\}$

```
SELECT ename FROM emp
UNION
SELECT dname FROM dept;
```

# RANK 관련 함수

```
SELECT sal, ename,  
       RANK() OVER (ORDER BY sal DESC) AS rank,  
       DENSE_RANK() OVER (ORDER BY sal DESC) AS dense_rank,  
       ROW_NUMBER() OVER (ORDER BY sal DESC) AS row_number,  
       rownum AS "rownum"  
FROM emp;
```



SAL	ENAME	RANK	DENSE_RANK	ROW_NUMBER	rownum
5000	KING	1	1	1	9
3000	FORD	2	2	2	13
3000	SCOTT	2	2	3	8
2975	JONES	4	3	4	4
2850	BLAKE	5	4	5	6
2450	CLARK	6	5	6	7
1600	ALLEN	7	6	7	2
1500	TURNER	8	7	8	10
1300	MILLER	9	8	9	14
1250	WARD	10	9	10	3
1250	MARTIN	10	9	11	5
1100	ADAMS	12	10	12	11
950	JAMES	13	11	13	12
800	SMITH	14	12	14	1

# 학습 마무리

---

- SELECT (II)
  - Join: 둘 이상의 테이블 결합
  - GROUP BY & Aggregation
  - Subquery