**JOIN**

1. 교집합
2. EQUI(등가) 조인

* 여러 개 릴레이션 사용해 새로운 릴레이션을 만드는 과정
* 조인의 가장 기본은 교집합을 만드는 것
* 두 테이블 간 일치하는 것 조인
* EQUI 조인은 “=”을 사용해서 두 개의 테이블을 연결

SELECT \* FROM EMP, DEPT

WHERE EMP.DEPTNO = DEPT.DEPTNO;

1. INNER JOIN

* EQUI 조인과 마찬가지로 ISO표준 SQL로, ON문을 사용해서 테이블 연결

SELECT \* FROM EMP INNER JOIN DEPT

ON EMP.DEPTNO = DEPT.DEPTNO;

* EQUI 조인을 한 후 실행계획을 확인해서 내부적 두 테이블을 어떻게 연결했는지 확인가능
* 전체 테이블을 읽고 해시함수를 사용해서 두 테이블을 연결한 것

<해시조인이란>

* 해시함수는 테이블을 해시메모리에 올리고 함수로써 연결한 것
* 해시조인은 EQUI조인만 사용가능
* 옵티마이저 파트에 Nested Loop, Sort merge와 같은 JOIN을 행하는 방법 중 하나

1. INTERSECT JOIN

* 두개의 테이블에서 교집합을 조회한다
* 두개 테이블에서 공통된 값을 조회한다

SELECT DEPTNO FROM EMP

INTERSECT

SELECT DEPTNO FROM DEPT;

2. 비등가조인

- 등가조인은 특정열에 일치한 데이터를 기준으로 출력행을 선정하는 방식이고 이 외의 조인은 비등가조인

- 두 개의 테이블 간 조인하는 경우 “=”를 사용하지 않고 “>”, “<”, “>=”, “<=”, BETWEEN 등을 사용

SELECT EMPLOYEE ID, FIRST\_NAME, SALARY, GRADE

FROM EMPLOYEES, SALARYGRADE

WHERE SALARY BETWEEN LOSALARY AND HISALARY;

3. OUTER JOIN

- 교집합(EQUI JOIN) + 한쪽테이블에만 있는 데이터도 포함

- FULL OUTER JOIN은 LEFT + RIGHT JOIN을 모두하는 것

- ORACLE에서는 OUTER JOIN을 할 때 “(+)”를 사용할 수 있음

SELECT \* FROM DEPT, EMP

WHERE EMP.DEPTNO (+) = DEPT.DEPTNO;

LEFT / RIGHT OUTER JOIN

SELECT \* FROM DEPT LEFT(RIGHT) OUTER JOIN EMP

WHERE EMP.DEPTNO = DEPT.DEPTNO;

4. CROSS JOIN

- 조인 조건구 없이 2개테이블을 하나로 조인

- 카테시안 곱이 발생한다. EX) 14개 행과 4개 행의 테이블을 조인하면 56개 행이 조회됨

SELECT \* FROM EMP CROSS JOIN DEPT;

QUIZ. EMP테이블에 10개 행이 있고 DEPT 테이블에 행이 2개 있을 때 다음의 SQL을 실행하면 조회되는 행의 수는?

SELECT \* FROM EMP, DEPT

20 40 80 160

5. UNION을 통한 합집합

(1) UNION

- 두개 테이블을 하나로 만드는 연산

- 두 테이블의 칼럼수와 데이터형식이 일치해야한다. 다르면 오류발생

- 두개의 테이블을 합치면서 중복 제거 > 정렬과정을 발생시킨다

SELECT DEPTNO FROM EMP

UNION

SELECT DEPTNO FROM DEPT;

WHERE EMP.DEPTNO (+) = DEPT.DEPTNO;

(2) UNION ALL

- 두 개의 테이블을 합치지만 중복제거하거나 정렬과정은 없음

SELECT DEPTNO FROM EMP

UNION ALL

SELECT DEPTNO FROM DEPT;

WHERE EMP.DEPTNO (+) = DEPT.DEPTNO;

6. 차집합 MINUS

-MINUS연산은 두개의 테이블에서 차집합을 조회.

- MS-SQL은 EXCEPT로 사용

SELECT DEPTNO FROM EMP

MINUS

SELECT DEPTNO FROM DEPT;

WHERE EMP.DEPTNO (+) = DEPT.DEPTNO;

**계층형 조회(Connect by)**

* Oracle에서 지원하는 것.
* 계층형데이터는 무엇 : 상위/하위 구조가 있는 데이터 EX) 대학-학과-과목 / 부장-차장-과장-대리-사원
* 시작조건 : START WITH / 조인조건 : CONNECT BY PRIOR

CONNECT BY 키워드

* LEVEL / CONNECT\_BY\_ROOT / CONNECT\_BY\_ITSELF / SYS\_CONNECT\_BY\_PATH 등
* 문제풀이 예시 : <https://www.youtube.com/watch?v=U4MlOLK2E9M>

QUIZ. 계층형 조회에서 제일 처음에 탐색을 시작하는 시점은 어느 것인가?

START WITH 구 CONNECT BY 구 FROM 구 SELECT 구

**서브쿼리(Subquery)**

1. 메인쿼리와 서브쿼리

-서브쿼리는 SELECT 문 내에 다시 SELECT 문을 사용하는 SQL문

-서브쿼리 형태는 1. FROM 구에 사용하는 인라인 뷰(View) 2. SELECT 문에 사용하는 스칼라 서브쿼리 등이 있음

- WHERE구에 SELECT문을 사용하면 중첩 서브쿼리(Nested subquery)라고 한다

SELECT \*

FROM EMP

WHERE DEPTNO =

(SELECT DEPTNO FROM DEPT

WHERE DEPTNO=10);

<인라인 뷰>

SELECT \*

FROM (SELECT ROWNUM NUM, ENAME

FROM EMP) a

WHERE NUM < 5;

2. 단일행 서브쿼리와 다중행 서브쿼리

- 서브쿼리는 반환하는 행이 한 개 / 여러 개인 것으로 단일행/멀티행 서브쿼리로 분류

- 단일행 서브쿼리는 단 하나의 행만 반환하는 서브쿼리로 비교연산자(=,>,< 등)를 사용한다

- 다중행 서브쿼리는 여러 개의 행을 반환하는 것으로 IN, ANY, ALL, EXISTS를 사용

IN 메인쿼리의 비교조건이 sub쿼리중 하나만 동일하면 참

<IN>

SELECT \* FROM EMP, DEPT

WHERE EMP.DEPTNO=DEPT.DEPTNO

AND EMP.EMPNO

IN (SELECT EMPNO FROM EMP

WHERE SAL > 2000) ;

ALL 메인쿼리와 서브쿼리 결과가 모두 동일하면 참

<ALL>

SELECT \* FROM EMP

WHERE DEPTNO <= ALL (20,30);

ANY 메인쿼리와 비교조건이 서브쿼리의 결과중 하나 이상 동일하면 참

EXISTS 메인쿼리와 서브쿼리 결과가 하나라도 존재하면 참(TRUE)

<EXISTS>

SELECT ENAME, DNAME, SAL

FROM EMP, DEPT

WHERE EMP.DEPTNO=DEPT.DEPTNO

AND EXISTS (SELECT 1 FROM EMP

WHERE SAL > 2000) ;

<Tips> EXISTS와 JOIN/IN의 차이

* IN과 EXISTS는 비슷한 결과가 조회되지만 IN은 서브쿼리 결과를 모두 수행하고 EXISTS는 일치하는 결과가 있으면 더 이상 수행하지 않는다. 서브쿼리테이블 데이터량이 많으면 EXISTS가 성능상 유리
* EXISTS의 서브쿼리를 메인쿼리 JOIN으로 변경할 경우 중복되는 데이터가 나올 수 있음. JOIN을 사용하려면 메인쿼리와 서브쿼리 데이터가 KEY를 기준으로 1:1일 경우 가능하다.

3. 스칼라 서브쿼리

- SELECT 문에 들어가며 반드시 한 행과 한 칼럼만 반환하는 서브쿼리

- 여러행 반환되면 오류 발생

SELECT ENAME AS “이름”

SAL AS “급여”

(SELECT AVG(SAL)

FROM EMP) AS “평균급여”

FROM EMP

4. 연관(Correlated) 서브쿼리

- 서브쿼리 내에서 메인쿼리 내 데이터를 받아서 서브쿼리를 실행하는 것

SELECT ENAME

FROM EMP a

WHERE a.DEPTNO =

(SELECT DEPTNO FROM DEPT b

WHERE b.DEPTNO=a.DEPTNO)

QUIZ. 다음 중 다중행 서브쿼리와 관련이 없는 것은?

ANY ALL = IN

**그룹함수(Group Function)**

1. ROLLUP

-GROUP BY의 칼럼에 대해서 subtotal을 만들어줌

-ROLLUP 할때 GROUP BY구에 칼럼이 두개이상 오면 순서에 따라서 결과가 달라짐

Ex) GROUP BY ROLLUP(1,2)이면 GROUP BY 1,2 합계 + GROUP BY 1 합계

GROUP BY ROLLUP(1,2,3)이면 GROUP BY 1,2,3 합계 + GROUP BY 1,2 합계 + GROUP BY 1 합계

2. GROUPING 함수

- ROLLUP / CUBE / GROUPING SETS에서 생성되는 합계 구분하기 위해 만들어진 함수

- 집계인 행을 1로, 아닌 것을 0으로 반환

- DECODE 함수을 통해 합계, 전체합계 등으로 출력가능

3. GROUPING SETS함수

- GROUP BY의 순서와 관계없이 다양한 소계 만들 수 있음

EX) GROUP BY 1,2이면 GROUP BY 1 합계 + GROUP BY 2 합계 조회

4. CUBE 함수

- CUBE함수에 제시한 칼럼에 대해서 결합가능한 모든 집계를 계산한다

- 다차원 집계를 제공하여 다양하게 데이터 분석가능

- 조합할 수 있는 모든 경우의 수가 모두 조합

EX) GROUP BY CUBE(1,2)면 GROUP BY 1,2 합계 + GROUP BY 1 합계 + GROUP BY 2합계

* 롤업은 GROUP BY 1,2 합계 + GROUP BY 1 합계 까지

QUIZ. 그룹함수(Group Function) 중에서 결합 가능한 모든 집계를 계산하는 것은?

ROLLUP GROUPING SETS GROUPING CUBE

**윈도우함수(Window Function)**

1. 윈도우함수

* 행과 행 간의 관계를 정의하기 위해 제공되는 함수
* 순위, 합계, 평균, 행위치 등 조작가능

SELECT **WINDOW\_FUNCTION(ARGUMENTS)**

**OVER(PARTITION BY 칼럼**

**ORDER BY WINDOWING절)**

FROM 테이블명;

* 구조 : ARGUMENTS / PARTITION BY / ORDER BY / WINDOWING
* WINDOWING : ROWS / RANGE / BETWEEM ~ AND / UNBOUNDED PRECEDING / UNBOUNDED FOLLOWING / CURRENT ROW

1. 순위 함수(RANK)

* 특정 항목과 파티션에 대해 순위를 계산할 수 있는 함수 제공
* RANK, DENSE\_RANK(동일순위 하나로 간주), ROW\_NUMBER(동일순위도 순위구분)
* RANK() OVER (PARTITION BY A / ORDER BY B DESC) : A로 파티션 만들고 B별 순위

1. 집계 함수(AGGREGATE)

* 윈도우 함수를 제공한다 / SUM, AVG, COUNT, MIN, MAX

EX) SUM(SAL) OVER (PARTITION BY MGR) SUM\_MGR

1. 행 순서 관련 함수

* FIRST\_VALUE / LAST\_VALUE / LAG / LEAD

1. 비율 관련 함수

* CUME\_DIST / PERCENT\_RANK / NTILE / RATIO\_TO\_REPORT

**테이블 파티션**

* 파티션은 대용량의 테이블을 여러 개의 데이터 파일에 분리해서 저장하는 것
* 테이블의 데이터가 물리적으로 분리된 데이터파일에 저장되면 입력,수정,삭제,조회성능이 향상됨
* 파티션은 각각의 파티션 별로 독립적으로 관리될 수 있음. 파티션별로 백업하고 복구가 가능하면 파티션전용 인덱스 생성도 가능
* 파티션은 ORACLE DB의 논리적 관리단위인 테이블 스페이스 간에 이동이 가능
* 데이터를 조회할 때 데이터의 범위를 줄여서 성능을 향상시킴

1. RANGE 파티션

- 테이블의 칼럼 중에서 값의 범위를 기준으로 여러 개의 파티션으로 데이터를 나누어 저장

2. LIST 파티션

- 특정 값을 기준으로 분할하는 방법

3. HASH 파티션

- 데이터베이스 관리시스템이 내부적으로 해시함수를 사용해서 데이터를 분할

- 결과적으로 시스템이 알아서 분할 관리하는 것

- 이외에도 COMPOSITE 파티션이 있는데 여러 개의 파티션 기법을 조합해서 사용하는 것

QUIZ. 파티션 기법 중에서 특정 값을 기준으로 데이터를 파티션하는 방법은?

Range 파티션 List 파티션 Hash 파티션 Composite 파티션

(5) 파티션 인덱스

- 파티션 인덱스는 4가지 유형의 인덱스 제공.

GLOBAL Index : 여러 개의 파티션에서 하나의 인덱스 사용

LOCAL Index : 해당 파티션 별로 각자의 인덱스를 사용

PREFIXED Index : 파티션 키와 인덱스 키가 동일

NON PREFIXED Index : 파티션 키와 인덱스 키가 다름