create user scott\_join identified by tiger;

grant all privileges to scott\_join;

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

조인(JOIN) 이란

- 2개 이상의 테이블에 있는 컬럼들을 합쳐서 가상의 테이블을 만들어 조회하는 방식을 말한다.

- 소스테이블 : 내가 먼저 읽어야 한다고 생각하는 테이블 (emp)

- 타겟테이블 : 소스를 읽은 후 소스에 조인할 대상이 되는 테이블 (dept)

- 각 테이블을 어떻게 합칠지를 표현하는 것을 조인 연산이라고 한다.

- 조인 연산에 따른 조인종류

- Equi join(PK=FK) , non-equi join(FK>=PK 와 같은 연산)

- 조인의 종류

- Inner Join (소스테이블에서 FK가 되는 컬럼이 null이 아닌 행만 조인)

- 양쪽 테이블에서 조인 조건을 만족하는 행들만 합친다.

- Outer Join (소스테이블에서 FK가 되는 컬럼이 null이어도 없는 컬럼의 값을 빈 상태로 조인)

- 한쪽 테이블의 행들을 모두 사용하고 다른 쪽 테이블은 조인 조건을 만족하는 행만 합친다. 조인조건을 만족하는 행이 없는 경우 NULL을 합친다.

- 종류 : Left Outer Join(왼쪽 소스테이블 오른쪽 타겟테이블), Right Outer Join(오른쪽 소스테이블 왼쪽 타겟테이블), Full Outer Join

- Cross Join (어떤 행을 합칠 것인지 명시하지 않음(ex. dept.dept\_id=emp.dept\_id 이런 과정X)

- 두 테이블의 곱집합을 반환한다.

- 카티션 곱: 하나의 행에 여러 다른 행이 붙는 경우

- 총 emp의 행 수 \* dept의 행 수의 결과가 나온다.

- 조인 문법

- ANSI 조인 문법

- 표준 SQL 문법

- 오라클은 9i 부터 지원.

- 오라클 조인 문법

- 오라클 전용 문법이며 다른 DBMS는 지원하지 않는다.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- inner join : ANSI 조인 구문

FROM 테이블a INNER JOIN 테이블b ON 조인조건

- inner는 생략 할 수 있다.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

-- 직원의 ID(emp.emp\_id), 이름(emp.emp\_name), 입사년도(emp.hire\_date), 소속부서이름(dept.dept\_name)을 조회

select emp.emp\_id, emp.emp\_name, emp.hire\_date, dept.dept\_name

--select \*

from emp join dept on emp.dept\_id=dept.dept\_id;

--emp e : e 테이블이름의 별칭(alias), dept d : d 테이블이름의 별칭

select e.emp\_id, e.emp\_name, e.hire\_date, d.dept\_name from emp e inner join dept d on e.dept\_id=d.dept\_id;

-- 직원의 ID(emp.emp\_id)가 100인 직원의 직원\_ID(emp.emp\_id), 이름(emp.emp\_name), 입사년도(emp.hire\_date), 소속부서이름(dept.dept\_name)을 조회.

select e.emp\_id, e.emp\_name, e.hire\_date, d.dept\_name

from emp e join dept d on e.dept\_id = d.dept\_id

where e.emp\_id=100;

-- 직원\_ID(emp.emp\_id), 이름(emp.emp\_name), 급여(emp.salary), 담당업무명(job.job\_title), 소속부서이름(dept.dept\_name)을 조회

select e.emp\_id, e.emp\_name, e.salary, j.job\_title, d.dept\_name

from emp e join job j on e.job\_id=j.job\_id

join dept d on e.dept\_id=d.dept\_id;

-- 부서\_ID(dept.dept\_id)가 30인 부서의 이름(dept.dept\_name), 위치(dept.loc), 그 부서에 소속된 직원의 이름(emp.emp\_name)을 조회.

select d.dept\_name, d.loc, e.emp\_name

from dept d join emp e on d.dept\_id=e.dept\_id

where d.dept\_id = 30;

-- 직원의 ID(emp.emp\_id), 이름(emp.emp\_name), 급여(emp.salary), 급여등급(salary\_grade.grade) 를 조회. 급여 등급 오름차순으로 정렬

select e.emp\_id, e.emp\_name, e.salary, s.grade||'등급'

from emp e join salary\_grade s on e.salary between s.low\_sal and s.high\_sal;

--TODO 200번대(200 ~ 299) 직원 ID(emp.emp\_id)를 가진 직원들의 직원\_ID(emp.emp\_id), 이름(emp.emp\_name), 급여(emp.salary),

-- 소속부서이름(dept.dept\_name), 부서위치(dept.loc)를 조회. 직원\_ID의 오름차순으로 정렬.

select e.emp\_id, e.emp\_name, e.salary, d.dept\_name, d.loc

from emp e join dept d on e.dept\_id=d.dept\_id

where e.emp\_id between 200 and 299

order by e.emp\_id;

--TODO 업무(emp.job\_id)가 'FI\_ACCOUNT'인 직원의 ID(emp.emp\_id), 이름(emp.emp\_name), 업무(emp.job\_id),

-- 소속부서이름(dept.dept\_name), 부서위치(dept.loc)를 조회. 직원\_ID의 오름차순으로 정렬.

select e.emp\_id, e.emp\_name, e.job\_id, d.dept\_name, d.loc

from emp e join dept d on e.dept\_id=d.dept\_id

where e.job\_id='FI\_ACCOUNT'

order by e.emp\_id;

--TODO 커미션비율(emp.comm\_pct)이 있는 직원들의 직원\_ID(emp.emp\_id), 이름(emp.emp\_name), 급여(emp.salary), 커미션비율(emp.comm\_pct),

-- 소속부서이름(dept.dept\_name), 부서위치(dept.loc)를 조회. 직원\_ID의 오름차순으로 정렬.

select e.emp\_id, e.emp\_name, e.salary, e.comm\_pct, d.dept\_name, d.loc

from emp e join dept d on e.dept\_id=d.dept\_id

where e.comm\_pct is not null

order by e.emp\_id;

--TODO 'New York'에 위치한(dept.loc) 부서의 부서\_ID(dept.dept\_id), 부서이름(dept.dept\_name), 위치(dept.loc),

-- 그 부서에 소속된 직원\_ID(emp.emp\_id), 직원 이름(emp.emp\_name), 업무(emp.job\_id)를 조회. 부서\_ID 의 오름차순으로 정렬.

select d.dept\_id, d.dept\_name, d.loc, e.emp\_id, e.emp\_name, e.job\_id

from dept d join emp e on d.dept\_id=e.dept\_id

where d.loc='New York'

order by d.dept\_id;

--TODO 직원\_ID(emp.emp\_id), 이름(emp.emp\_name), 업무\_ID(emp.job\_id), 업무명(job.job\_title) 를 조회.

select e.emp\_id, e.emp\_name, e.job\_id, j.job\_title

from emp e join job j on e.job\_id=j.job\_id;

-- TODO: 직원 ID 가 200 인 직원의 직원\_ID(emp.emp\_id), 이름(emp.emp\_name), 급여(emp.salary),

-- 담당업무명(job.job\_title), 소속부서이름(dept.dept\_name)을 조회

select e.emp\_id, e.emp\_name, e.salary, j.job\_title, d.dept\_name

from emp e join job j on e.job\_id=j.job\_id

join dept d on e.dept\_id=d.dept\_id

where e.emp\_id=200;

-- TODO: 'Shipping' 부서의 부서명(dept.dept\_name), 위치(dept.loc), 소속 직원의 이름(emp.emp\_name), 업무명(job.job\_title)을 조회.

-- 직원이름 내림차순으로 정렬

select d.dept\_name, d.loc, e.emp\_name, j.job\_title

from dept d join emp e on d.dept\_id=e.dept\_id

join job j on e.job\_id=j.job\_id

where d.dept\_name = 'Shipping'

order by e.emp\_name desc;

-- TODO: 'San Francisco' 에 근무(dept.loc)하는 직원의 id(emp.emp\_id), 이름(emp.emp\_name), 입사일(emp.hire\_date)를 조회

-- 입사일은 'yyyy-mm-dd' 형식으로 출력

select e.emp\_id, e.emp\_name, to\_char(e.hire\_date,'yyyy-mm-dd') hire\_date

from emp e join dept d on d.dept\_id=e.dept\_id

where d.loc= 'San Francisco';

-- TODO 부서별 급여(salary)의 평균을 조회. 부서이름(dept.dept\_name)과 급여평균을 출력. 급여 평균이 높은 순서로 정렬.

-- 급여는 , 단위구분자와 $ 를 붙여 출력.

select d.dept\_name, to\_char(avg(e.salary),'$9,999,999')"평균급여"

from dept d join emp e on d.dept\_id=e.dept\_id

group by d.dept\_name

order by avg(e.salary) desc;

--order by 2 desc; --이 때 평균급여를 문자열로 비교하기 때문에 fm을 붙여 공백을 없앨 경우 정렬이 제대로 되지 않는 문제가 생긴다.

--TODO 직원의 ID(emp.emp\_id), 이름(emp.emp\_name), 업무명(job.job\_title), 급여(emp.salary),

-- 급여등급(salary\_grade.grade), 소속부서명(dept.dept\_name)을 조회. 등급 내림차순으로 정렬

select e.emp\_id, e.emp\_name, j.job\_title, e.salary, s.grade, d.dept\_name

from emp e join job j on e.job\_id=j.job\_id

join salary\_grade s on e.salary between s.low\_sal and s.high\_sal

join dept d on e.dept\_id=d.dept\_id

order by 5 desc;

--TODO 급여등급이(salary\_grade.grade) 1인 직원이 소속된 부서명(dept.dept\_name)과 등급1인 직원의 수를 조회. 직원수가 많은 부서 순서대로 정렬.

/\*select d.dept\_name, count(\*) 직원수

from dept d join emp e on d.dept\_id=e.dept\_id

join salary\_grade s on e.salary between s.low\_sal and s.high\_sal

group by d.dept\_name, s.grade

having s.grade=1

order by 2 desc;\*/

---------------------------------------------------------------------------------

select d.dept\_name, count(\*) 직원수

from dept d join emp e on d.dept\_id=e.dept\_id

join salary\_grade s on e.salary between s.low\_sal and s.high\_sal

where s.grade=1

group by d.dept\_name

--group by rollup(dept\_name) : 총 1등급인 직원의 수

--having count(\*)>=4 : 1등급인 직원의 수가 4명 이상인 부서명

order by 2 desc;