조인

조인의 필요성

❖ 데이터의 일관성

- 데이터를 여러 테이블에 중복해서 저장하는 경우 일관성 문제 발생
- 데이터가 중복되지 않도록 2개 이상의 테이블에 나누어 저장
- 원하는 정보를 얻으려면 여러 개의 테이블을 조인

- 사원의 부서명을 함께 출력
  - EMPLOYEES 테이블의 DEPARTMENT_ID와 DEPARTMENTS 테이블의 DEPARTMENT_ID 공통값으로 두 테이블을 연결

조인의 필요성

❖ 두개 테이블의 조인 필요성

- LAST_NAME이 ‘Lee’인 사원의 부서명 조회

```
SELECT department_id
FROM employees
WHERE last_name = 'Lee';
→ 80
```

- 부서 번호 80인 부서 조회

```
SELECT *
FROM departments
WHERE department_id=80;

→ 80    Sales    145      2500
```

Cross Join

❖ Cross Join

- 2개 이상의 테이블이 조인될 때 WHERE절에 의해 공통되는 컬럼에 의한 결합이 발생하지 않은 경우
- 테이블 전체 컬럼이 조인된다
- 로우 수: 테이블 1의 로우 수 X 테이블 2의 로우 수

```
SELECT *
FROM employees, departments;
```

❖ 조인의 종류

- 조인 결과가 의미를 갖으려면 조인할 때 조건을 지정

종류	설명
Equi Join	동일 컬럼을 기준으로 조인
Non-Equi Join	동일 컬럼이 없이 다른 조건을 사용하여 조인
Outer Join	조인 조건에 만족하지 않는 로우도 나타냄
Self Join	한 테이블 내에서 조인

Equi Join

❖ Equi Join

- 가장 많이 사용하는 조인 방법
- 조인 대상이 되는 두 테이블에서 공통적으로 존재하는 컬럼의 값이 일치되는 행을 연결하여 결과를 생성하는 방법

```
SELECT *
FROM employees, departments;
WHERE employees.department_id = departments.department_id;
```

- 조인 테이블의 특정 필드만 출력

```
SELECT first_name, last_name, department_name
FROM employees, departments
WHERE employees.department_id = departments.department_id;
```

Equi Join

❖ 별칭 사용

- 필드명이 충돌나는 경우 에러 발생

```
SELECT first_name, last_name, department_id, department_name
FROM employees, departments
WHERE employees.department_id = departments.department_id;
```

- 테이블 명.필드명으로 표기

```
SELECT employees.first_name, employees.last_name,
        departments.department_id, departments.department_name
FROM employees, departments
WHERE employees.department_id = departments.department_id;
```

Equi Join

❖ 별칭 사용

- 테이블의 별칭 사용 가능

```
SELECT emp.first_name, emp.last_name,
         dep.department_id, dep.department_name
FROM employees emp, departments dep
WHERE emp.department_id = dep.department_id;
```

Equi Join

❖ 실습

- 조인을 사용하여 ‘Purchasing’ 부서에서 근무하는 사원의 이름과 급여를 출력하시오.

- 조인을 사용하여 ‘Accounting’ 부서 소속 사원의 이름과 입사일을 출력하세요

- 각 부서의 관리자에 대해서 부서명, 사원명, 입사일을 출력하세요

Non-Equi Join

❖ Non-Equi Join

- 조인 조건의 특정 범위 내에 있는지 조사하기 위해 WHERE절의 조인 조건으로 =연산자와 비교 연산자를 사용한다

- 2007년 입사사원 이름과 부서명을 출력하세요
  (부서별로 오름차순으로 출력)

Self Join

❖ Self Join

- 하나의 테이블 내에서 조인을 해야만 자료를 얻을 수 있는 경우
- 자기 자신과 조인을 맺는 것

```
SELECT first_name, last_name, manager_id
FROM employees;
```

- 매니저의 상세 정보를 조회하려면 자기 자신에 대한 조인 필요
- 반드시 테이블의 별칭을 사용

```
SELECT emp.first_name || ' ' || emp.last_name || '의 매니저는 ' ||
        mgr.first_name || ' ' || mgr.last_name || '입니다.'
FROM employees emp, employees mgr
WHERE emp.manager_id=mgr.employee_id;
```

Self Join

❖ 실습

- 매니저가 King 사원들의 이름과 직급(JOB_ID)을 출력하시오.

- Jones와 동일한 부서에서 근무하는 사원의 이름을 출력하세요.

Self Join

```
SELECT  work.first_name, work.last_name, work.job_id
FROM    employees work, employees manager
WHERE   work.manager_id = manager.employee_id
        AND manager.last_name='King';
```

```
SELECT  work.last_name, friend.last_name
FROM    employees work, employees friend
WHERE   work.department_id = friend.department_id
        AND work.last_name ='Jones'
        AND friend.last_name <> 'Jones'
```

Outer Join

❖ Outer Join

- 2개 이상의 테이블이 조인될 때 어느 한 쪽의 테이블에는 해당하는 데이터가 존재하는데 다른 쪽 데이블에는 데이터가 존재하지 않는 경우(NULL) 그 데이터는 출력되지 않는 문제를 해결하기 위해서 사용되는 조인 기법

- Self Join시 Steven King은 결과에서 빠짐
  - MANAGER_ID가 NULL이기 때문에 Join시 빠짐
- 조인 조건에 만족하지 못하더라도 해당 로우를 출력 하고 싶을 때 외부 조인 사용
- 해당 필드에 (+) 표시

```
SELECT emp.first_name || ' ' || emp.last_name || '의 매니저는 ' ||
        mgr.first_name || ' ' || mgr.last_name || '입니다.'
FROM employees emp, employees mgr
WHERE emp.manager_id=mgr.employee_id(+);
```

Outer Join

❖ **사원 테이블과 부서 테이블을 조인하여 사원명과 부서번호, 부서명 출력**

```
SELECT e.first_name, e.last_name, d.department_id, d.department_name
FROM employees e, departments d
WHERE e.department_id = d.department_id
ORDER BY d.department_id;

→ 출력의 개수 106개, 사원수는 107 … 한 개는 어디로?
```

```
SELECT e.first_name, e.last_name, d.department_id, d.department_name
FROM employees e, departments d
WHERE e.department_id = d.department_id(+)
ORDER BY d.department_id;
```