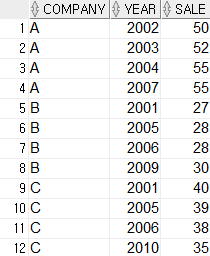
**SQL 반복문**

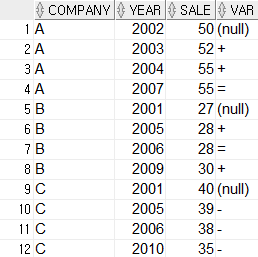
* SQL 에는 반복문이 존재하지 않는다.
* 반복문을 사용하지 않는 것이 성능에 좋다고 판단했기 때문이다.
* 하지만 반복문에 익숙해진 개발자는 호스트 언어(Java, C, C++, …)에서 반복 처리를 구현한다.
* **반복문을 사용하면 레코드의 집합에 접근하지 못 하고, 레코드마다 작은 SQL을 사용해서 접근하게 되므로 성능이 좋지 않으므로 반복문을 사용하지 않는 방법을 알아보자.**
* **Sales TABLE**



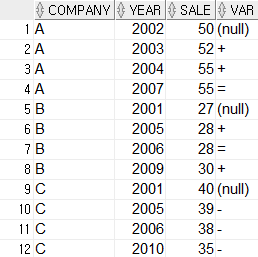
* **Sales2 TABLE**



* **문제1] Sales테이블은 각 기업의 회계연도별 매출을 기록한다. 연도가 연속되지 않는다. 이 데이터를 사용해 특정기업의 매출 변화를 조사할 것 이다. 그리고 결과는 var 필드를 추가한 Sales2 테이블에 등록한다. (Cursor, Procedure, 반복문 등 사용.)**
* Var 필드는 다음과 같은 규칙에 따라 결정된다.
  + 이전 데이터가 없는 경우: NULL
  + 이전 데이터보다 매출이 오른 경우: +
  + 이전 데이터보다 매출이 내린 경우: -
  + 이전 데이터와 매출이 같은 경우: =
* **Sale2 테이블은 다음과 같은 데이터가 들어간다.**



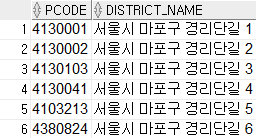
* **문제2] [문제1]을 반복문을 사용하지 않고, SQL을 사용해서 풀어라. (CASE식과 윈도우 함수사용)**
* 다음과 같은 테이블이 나오면 된다.



* **문제3] 윈도우 함수로 ‘직전 회사명’과 ‘직전 매출’ 검색, 데이터테이블은 문제2번의 테이블을 사용**
* 다음과 같은 테이블이 나와야 한다.



* **문제 4] 아래 테이블에 저장된 우편번호의 집합에서, 입력 받은 우편 번호('4130033’)와 가장 가까운 지역의 우편번호를 검색해보자. 왼쪽부터 자릿수가 많이 일치할수록 가까운 우편번호다.**



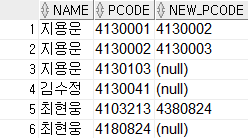
* 다음과 같은 결과가 나오면 된다.



* **문제 5] [문제4]를 1회의 테이블 스캔, 즉 1회의 SELECT문으로 해결하지 못 했다면 1회의 테이블 스캔으로 문제를 해라.**
* 다음과 같은 테이블이 나오면 된다.

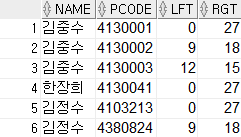


* **문제6] 아래 테이블에서 '지용운’씨가 가장 오래 전에 살았던 주소를 찾아라.**
* 이 테이블에서 현재 주소를 등록할 때는 현재 주소의 우편번호만 등록하고, 이사하는 곳의 우편번호는 NULL 로 한다.
* 이사를 하게 되면 NULL값 이던 필드에 이사하는 곳의 우편번호를 넣어서 레코드를 수정한다. 그리고
* 새로운 우편번호와 ‘NULL 값을 갖는 새로운 우편번호 필드’를 갖는 레코드를 생성한다.
* 우편번호를 키로 삼아 데이터를 줄줄이 연결하고 있다(포인터 체인).



* 만약 ‘지용운’씨가 가장 오래 전에 살았던 주소를 검색한다면 답은 '4130001’이다.
* 이것을 찾으려면 현재 주소에서 출발해서 차근차근 이전 주소를 찾아야 할 것 이다.
* 문제는 몇 번을 따라 올라가야만 가장 오래된 주소를 찾을 수 있을 것인지 사전에는 알 수 없다는 점이다.
* **SQL에서 계층 구조를 찾는 방법 중 하나는 재귀 공통 테이블 식을 사용하는 방법이다.**

1. **중첩 집합 모델**



* 이 방법의 포인트는 **각 레코드의 데이터를 집합(원)으로 보고**, 계층 구조를 집합의 중첩 관계로 나타낸다는 것 이다.
* 우편번호의 데이터를 수치선 상에 존재하는 원으로 생각한다.
* **Lft와 Rft는 원의 왼쪽 끝과 오른쪽 끝에 위치하는 좌표를 나타낸다.**
* 그리고 이사할 때 마다 새로운 우편번호가 이전의 우편번호 '안에’ 포함되는 형태로 추가 한다.
* ‘김중수’씨의 가장 오래된 주소는 가장 바깥쪽에 있는 원을 찾기만 하면 되므로 굉장히 간단한 SQL 구문으로 찾을 수 있다.
* NOT EXISTS를 사용하면 쉽게 구할 수 있다.
* **문제 7 ] '김중수’씨의 가장 오래된 주소를 찾아라.**
* **결과**

