**1. Join**

하나 이상의 테이블로부터 자료를 검색하기 위하여 조인을 사용합니다. 일반적으로Primary Key(이후 PK로 사용)와 Foreign Key(이후 FK로 사용)을 사용하여 Join하는 경우가 대부분이지만 때로는 논리적인 값들의 연관으로 Join하는 경우도 있습니다.

**1.1 Syntax**

|  |
| --- |
| **SELECT** table1.column1 [,table2.column2, . . . . .]  **FROM** table1, table2  **WHERE** table1.column1 = table2.column2; |

☞ Guidelines

1. WHERE 절에 조인 조건을 기술한다.
2. 테이블을 조인하는 SELECT 문장을 작성할 경우 명확성을 위하여 또는 데이터베이스의 Performance 향상을 위하여 열 이름 앞에 테이블 명을 붙인다.
3. 똑 같은 열 이름이 존재하는 테이블이 있을 경우는 반드시 열 이름 앞에 테이블 명을 붙인다.
4. n개의 테이블을 조인 하려면 최소한 n-1번의 조인 조건 문이 필요하다.

**1.2 Join의 종류**

|  |  |
| --- | --- |
| Join 방법 | 설 명 |
| Cartesian Product | 모든 가능한 행들의 Join |
| Equijoin | Join조건이 정확히 일치하는 경우 사용(일반적으로PK와 FK사용) |
| Non-Equijoin | Join조건이 정확히 일치하지 않는 경우에 사용(등급,학점) |
| Outer Join | Join조건이 정확히 일치하지 않는 경우에도 모든 행들을 출력 |
| Self Join | 하나의 테이블에서 행들을 Join하고자 할 경우에 사용 |
| Set Operators | 여러 개의 SELECT문장을 연결하여 작성한다. |

**1.3 Cartesian Product**

모든 가능한 행들의 Join으로 다음과 같은 경우에 발생한다.

1. 조인 조건이 생략된 경우
2. 조인 조건이 잘못된 경우
3. 첫번째 테이블의 모든 행이 두번째 테이블의 모든 행과 두번째 테이블의 모든 행이 첫번째 테이블의 모든 행과 조인되는 경우.
4. 양쪽 ROW의 개수를 곱한 결과

♣ 참고

Cartesian Product는 만은 수의 행을 생성하는 경향이 있고 결과도 거의 유용하지 못하다. 그러므로 모든 테이블로부터 모든 행을 조합할 필요가 없을 경우 WHERE절에 조인 조건을 명확히 기술하여야 한다.

문제1) EMP 테이블과 DEPT 테이블을 Cartesian Product하여 사원번호,이름,업무,부서번호,부서명,근무지를 출력하여라.

|  |
| --- |
| SQL> **SELECt empno,ename,job,dept.deptno,dname,loc**  2 **FROM dept,emp**  3 **ORDER BY empno;**  EMPNO ENAME JOB DEPTNO DNAME LOC  --------- ---------- --------- --------- -------------- -------------  7369 SMITH CLERK 10 ACCOUNTING NEW YORK  7369 SMITH CLERK 20 RESEARCH DALLAS  7369 SMITH CLERK 30 SALES CHICAGO  7369 SMITH CLERK 40 OPERATIONS BOSTON  7499 ALLEN SALESMAN 10 ACCOUNTING NEW YORK  7499 ALLEN SALESMAN 20 RESEARCH DALLAS  7499 ALLEN SALESMAN 30 SALES CHICAGO  7499 ALLEN SALESMAN 40 OPERATIONS BOSTON  7521 WARD SALESMAN 10 ACCOUNTING NEW YORK  7521 WARD SALESMAN 20 RESEARCH DALLAS  7521 WARD SALESMAN 30 SALES CHICAGO  7521 WARD SALESMAN 40 OPERATIONS BOSTON  7566 JONES MANAGER 10 ACCOUNTING NEW YORK  7566 JONES MANAGER 20 RESEARCH DALLAS  7566 JONES MANAGER 30 SALES CHICAGO  . . . . . . . . . .  56 rows selected. |

**1.4 Equijoin**

Equijoin 이란 조인 조건에서 “=”을 사용하여 값들이 정확하게 일치하는 경우에 사용하는 조인을 말합니다. 대부분 PK와 FK의 관계를 이용하여 조인 합니다. Equijoin은 다른 말로 단순 조인 또는 내부 조인 이라고도 합니다.

**1.4.1) Syntax**

|  |
| --- |
| **SELECT**  table1.column1 [,table2.column2, . . . . .]  **FROM** table1, table2  **WHERE** **table1.column1 = table2.column2**; |

table1.column1 조회할 자료가 있는 테이블과 열 이름을 기술

table1.column1=table2.column2 두 테이블들간에 논리적으로 연결하는 조인 조건 기술

**1.4.2) Equijoin으로 자료 검색**

1. SELECT절은 검색할 열 이름을 명시
2. FROM절은 데이터베이스가 Access해야 하는 두개의 테이블을 명시
3. WHERE절은 테이블의 조인 조건을 명시
4. 양쪽 테이블에 공통으로 존재하는 열 이름은 모호함을 피하기 위하여 열 이름 앞에 테이블명을 기술함

**1.4.3) Equijoin 의 방법**

종업원의 부서 이름을 결정하기 위해 EMP Table의 DEPTNO와 DEPT Table의 DEPTNO와 값을 비교하여야 합니다. EMP Table과 DEPT Table 사이의 관계는 양쪽 테이블의 DEPTNO열이 같아야 합니다. 이들이 PK와 FK로 연결되어 있습니다.

SQL> SELECT empno,ename,job,deptno SQL> SELECT deptno,dname,loc

2 FROM emp; 2 FROM dept;

EMPNO ENAME JOB DEPTNO DEPTNO DNAME LOC

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

7839 KING PRESIDENT 10 10 ACCOUNTING NEW YORK

7698 BLAKE MANAGER 30 20 RESEARCH DALLAS

7782 CLARK MANAGER 10 30 SALES CHICAGO

7566 JONES MANAGER 20 40 OPERATIONS BOSTON

7654 MARTIN SALESMAN 30

7499 ALLEN SALESMAN 30

. . . . . . . .

14 rows selected.

문제2) EMP 테이블에서 사원번호,이름,업무,EMP 테이블의 부서번호,DEPT 테이블의 부서번호,부서명,근무지를 출력하여라.

|  |
| --- |
| SQL> **SELECT empno,ename,job,emp.deptno,dept.deptno,dname,loc**  2 **FROM dept,emp**  3 **WHERE dept.deptno = emp.deptno**  4 **ORDER BY dept.deptno;**    EMPNO ENAME JOB DEPTNO DEPTNO DNAME LOC  --------- ---------- --------- --------- --------- -------------- -------------  7839 KING PRESIDENT 10 10 ACCOUNTING NEW YORK  7782 CLARK MANAGER 10 10 ACCOUNTING NEW YORK  7934 MILLER CLERK 10 10 ACCOUNTING NEW YORK  7566 JONES MANAGER 20 20 RESEARCH DALLAS  7788 SCOTT ANALYST 20 20 RESEARCH DALLAS  . . . . . . . . . .  14 rows selected. |

**1.4.4) Table에 Alias사용**

1. 테이블 별칭을 사용하여 긴 테이블 명을 간단하게 사용한다.
2. 테이블 이름 대신에 Alias를 사용한다.
3. SQL 코드를 적게 사용하여 코딩 시간이 절약되고 메모리를 보다 적게 사용한다.

|  |
| --- |
| SQL> **SELECT e.empno,e.ename,e.job,e.deptno,**  2 **d.deptno,d.dname,d.loc**  3 **FROM dept d,emp e**  4 **WHERE d.deptno = e.deptno**  5 **ORDER BY d.deptno;**  EMPNO ENAME JOB DEPTNO DEPTNO DNAME LOC  --------- ---------- --------- --------- --------- -------------- -------------  7839 KING PRESIDENT 10 10 ACCOUNTING NEW YORK  7782 CLARK MANAGER 10 10 ACCOUNTING NEW YORK  7934 MILLER CLERK 10 10 ACCOUNTING NEW YORK  7566 JONES MANAGER 20 20 RESEARCH DALLAS  7788 SCOTT ANALYST 20 20 RESEARCH DALLAS  7876 ADAMS CLERK 20 20 RESEARCH DALLAS  . . . . . . . . . . .  14 rows selected. |

☞ Guidelines

1. 테이블 Alias는 30자까지 사용 가능하지만 짧을수록 더 좋다.
2. FROM절에서 Alias가 사용되면 SELECT문 전체에서 사용 가능하다.
3. 테이블의 Alias에 가급적 의미를 부여
4. 테이블은 현재 SELECT문장에서만 유용

**1.4.5) AND연산자를 사용하여 추가적인 검색 조건**

조인 이외의 WHERE절에 추가적인 조건을 가질 수 있다.

문제3) SALESMAN 사원만 사원번호,이름,급여,부서명,근무지를 출력하여라

|  |
| --- |
| SQL> **SELECT e.empno,e.ename,e.sal,d.dname,d.loc**  2 **FROM dept d,emp e**  3 **WHERE d.deptno = e.deptno AND e.job = 'SALESMAN';**    EMPNO ENAME SAL DNAME LOC  --------- ---------- --------- -------------- ----------  7654 MARTIN 1250 SALES CHICAGO  7499 ALLEN 1600 SALES CHICAGO  7844 TURNER 1500 SALES CHICAGO  7521 WARD 1250 SALES CHICAGO |

**1.4.6) 두개 이상의 테이블 조인**

때로는 두개 이상의 테이블을 조인 할 경우가 있다.

문제4) 고객의 TKB SPORT SHOP의 이름,주문처,항목수,각 항목의 합계,각 주문의 합계를 출력하여라.

|  |  |  |
| --- | --- | --- |
| SQL> **SELECT name,custid**  2  **FROM customer;**  NAME CUSTID  ---------------- -------  JOCKSPORTS 100  TKB SPORT SHOP 101  VOLLYRITE 102  . . . . . . . . .  9 rows selected. | SQL> **SELECT ordid,itemid**  2 **FROM item;**  ORDID ITEMID  --------- ---------  610 3  611 1  612 1  . . . . . . . . . .  64 rows selected. | SQL> **SELECT custid,ordid**  2 **FROM ord;**  CUSTID ORDID  --------- ---------  101 610  102 611  104 612  . . . . . . . . . .  21 rows selected. |
| SQL> **SELECT c.name,o.ordid,i.itemid,i.itemtot,o.total**  2 **FROM customer c,ord o,item i**  3 **WHERE c.custid = o.custid AND o.ordid = i.ordid**  4 **AND c.name = 'TKB SPORT SHOP';**    NAME ORDID ITEMID ITEMTOT TOTAL  -------------------- --------- --------- --------- ---------  TKB SPORT SHOP 610 3 58 101.4  TKB SPORT SHOP 610 1 35 101.4  TKB SPORT SHOP 610 2 8.4 101.4 | | |

**1.5 Non-Equijoin**

EMP와 SALGRADE 사이의 관련성은 EMP 테이블의 어떠한 column도 직접적으로 SALGRADE 테이블의 한 column에 상응하지 않기 때문에 Non-Equijoin이다. 두 테이블 사이의 관련성은 EMP 테이블의 SAL열이 SALGRADE 테이블의 LOSAL과 HISAL열 사이에 있다는 것이다. 조인 조건은 등등(=) 이외의 연산자(BETWEEN ~ AND ~)를 갖는다.

SQL> **SELECT empno,ename,sal** SQL> **SELECT grade,losal,hisal**

2 **FROM emp;** 2 **FROM salgrade;**

EMPNO ENAME SAL GRADE LOSAL HISAL

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

7839 KING 5000 1 700 1200

7698 BLAKE 2850 2 1201 1400

7782 CLARK 2450 3 1401 2000

7566 JONES 2975 4 2001 3000

7654 MARTIN 1250 5 3001 9999

. . . . . . . . .

14 rows selected.

문제5) EMP 테이블에서 사원번호,이름,업무,급여,급여의 등급,하한값,상한값을 출력하여라.

|  |
| --- |
| SQL> **SELECT e.empno,e.ename,e.job,e.sal,s.grade,s.losal,s.hisal**  2 **FROM salgrade s,emp e**  3 **WHERE e.sal BETWEEN s.losal AND s.hisal AND e.deptno = 10;**  EMPNO ENAME JOB SAL GRADE LOSAL HISAL  --------- ---------- --------- --------- --------- --------- ---------  7839 KING PRESIDENT 5000 5 3001 9999  7782 CLARK MANAGER 2450 4 2001 3000  7934 MILLER CLERK 1300 2 1201 1400 |

♣ 참고

위 질의가 실행될 때 한번만 조인되는 것을 알 수 있다. 이에 대한 두가지 이유가 있다.

1. SALGRADE 테이블에서 중복되는 등급을 포함하는 행이 없다.
2. EMP 테이블에 있는 SAL의 값은 SALGRADE 테이블에서 제공하는 값 범위에 있다.

☞ Guidelines

<= 및 >= 같은 다른 연산자를 사용 가능하나 BETWEEN이 가장 단순하다. 또한 테이블에 Alias를 사용하였는데 이는 모호성 때문이 아니라 성능 때문에 사용하였다. BETWEEN 사용시 하한값을 먼저 명시하고 상한값을 나중에 명시한다는 것을 명심하라.

**1.6 Outer Join**

행이 조인 조건을 만족하지 않으면, 행은 질의 결과에 나타나지 않을 것입니다.예를 들어 EMP와DEPT테이블의 equijoin조건에서 부서OPERATIONS(40번 부서)는 해당 부서에 아무도 없기 때문에 나타나지 않습니다. 이런 경우 모든 행을 전부 출력하고자 할 경우 Outer Join을 사용한다. 즉 정상적으로 조인 조건을 만족하지 못하는 행들을 보기 위해 Outer join을 사용한다.

SQL> **SELECT empno,ename,job,deptno**  SQL> **SELECT deptno,dname,loc**

2 **FROM emp; 2 FROM dept;**

EMPNO ENAME JOB DEPTNO DEPTNO DNAME LOC

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

7839 KING PRESIDENT 10 10 ACCOUNTING NEW YORK

7698 BLAKE MANAGER 30 20 RESEARCH DALLAS

7782 CLARK MANAGER 10 30 SALES CHICAGO

7566 JONES MANAGER 20 40 OPERATIONS BOSTON

7654 MARTIN SALESMAN 30

40번 부서는 조인시 조인되지 않는다.

. . . . . . . .

14 rows selected.

☞ Guidelines

1. 행인 조건을 만족하지 않을 시 해당 행은 질의 결과에 나타나지 않는다.
2. Outer join 연산자를 조인 조건에 사용시 조인 조건을 만족하지 않는 행들도 결과에 나타날 수 있다.
3. 연산자는 괄호로 묶인 플러스 기호(+)이며 조인 시킬 값이 없는 조인 측에 "(+)"를 위치 시킨다.
4. (+)연산자는 한 개 이상의 NULL 행을 생성하고 정보가 충분한 테이블의 한 개 이상의 행들이 이런 NULL 행에 조인된다.
5. Outer join 연산자는 표현식의 한 편에만 올 수 있다.
6. Outer join을 포함하는 조건은IN 연산자, OR 연산자를 사용하여 다른 하나의 조건에 연결될 수 없다.

**1.6.1) Syntax**

|  |
| --- |
| **SELECT**  table1.column1 [,table2.column2, . . . . .]  **FROM** table1, table2  **WHERE** table1.column1 = table2.column2**(+)**; |
| **SELECT** table1.column1 [,table2.column2, . . . . .]  **FROM** table1, table2  **WHERE** table1.column1**(+)** = table2.column2; |

table1.column 테이블을 함께 조인(또는 관련)시키는 조건입니다.

table2.column(+) (+)는 outer join기호입니다. WHERE절 조건의 양쪽이 아니라

어느 한쪽에 둘 수 있습니다. 즉 양측 모두에는 올 수 없습니다.

일치하는 행이 없는 테이블의 열 이름 뒤에 outer join연산자

를 사용합니다.

**1.6.2) Outer Join제약 사항**

1. Outer Join연산자는 정보가 부재하는 쪽의 표현식 한 쪽에만 둡니다. 다른 테이블의 어떠한 열과도 직접적으로 일치하는 것이 없는 한 테이블의 행을 리턴합니다.
2. Outer Join을 포함하는 조건은 IN연산자를 사용할 수 없고, OR연산자에 의해 다른 조건과 연결될 수 없습니다.

|  |
| --- |
| SQL> **select \* from emp,dept**  2 **where dept.deptno(+) = emp.deptno(+);**  where dept.deptno(+) = emp.deptno(+)  \*  ERROR at line 2:  ORA-01468: a predicate may reference only one outer-joined table  SQL> |

문제6) EMP 테이블과 DEPT 테이블에서 DEPT 테이블에 있는 모든 자료를 사원번호,이름,업무,EMP 테이블의 부서번호,DEPT 테이블의 부서번호,부서명,근무지를 출력하여라

|  |
| --- |
| SQL> **SELECT e.empno,e.ename,e.job,e.deptno,**  2 **d.deptno,d.dname,d.loc**  3 **FROM dept d,emp e**  4  **WHERE d.deptno = e.deptno(+);**    EMPNO ENAME JOB DEPTNO DEPTNO DNAME LOC  --------- ---------- --------- --------- --------- -------------- -------------  . . . . . . . . . . .  7900 JAMES CLERK 30 30 SALES CHICAGO  7521 WARD SALESMAN 30 30 SALES CHICAGO  40 OPERATIONS BOSTON  15 rows selected. |

**1.7 Self Join**

때때로 자체적으로 테이블을 조인할 필요가 있습니다. 각 종업원의 관리자 명을 알기 위해서 자체적으로 EMP테이블을 조인하는 것이 필요합니다.

1. ENAME열을 검사하여 EMP테이블에서 Blake를 검색한다.
2. MGR열을 검사하여 Blake에 대한 관리자 번호를 검색한다.(Blake 관리자 번호:7839)
3. ENAME열을 검사하여 EMPNO가 7839인 관리자를 검색한다.7839는 King이므로 Blake의 관리자는 King이다.

SQL> **SELECT empno,ename,mgr** SQL> **SELECT empno,ename,mgr**

2  **FROM emp;**  2 **FROM emp;**

EMPNO ENAME MGR EMPNO ENAME MGR

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

7839 KING 7839 KING

7698 BLAKE 7839 7698 BLAKE 7839

7782 CLARK 7839 7782 CLARK 7839

7566 JONES 7839 7566 JONES 7839

7654 MARTIN 7698 7654 MARTIN 7698

7499 ALLEN 7698 7499 ALLEN 7698

7844 TURNER 7698 7844 TURNER 7698

7900 JAMES 7698 7900 JAMES 7698

7521 WARD 7698 7521 WARD 7698

7902 FORD 7566 7902 FORD 7566

7369 SMITH 7902 7369 SMITH 7902

7788 SCOTT 7566 7788 SCOTT 7566

7876 ADAMS 7788 7876 ADAMS 7788

7934 MILLER 7782 7934 MILLER 7782

14 rows selected. 14 rows selected.

☞ Guidelines

1. Self join을 사용하여 한 테이블의 행들을 같은 테이블의 행들과 조인한다.
2. 같은 테이블에 대해 두 개의 alias를 작성(테이블 구분)함으로 FROM절에 두개의 테이블을 사용하는 것과 같이 한다.
3. Column에 대해서도 어떤 테이블에서 왔는지 반드시 Alias명을 기술하여야 한다.
4. 테이블 하나를 두개 또는 그 이상으로 Self join할 수 있다.

문제7) EMP 테이블에서 Self join하여 관리자를 출력하여라.

|  |
| --- |
| SQL> **SELECT worker.ename || '의 관리자는 ' || manager.ename || '이다'**  2 **FROM emp worker, emp manager**  3 **WHERE worker.mgr = manager.empno;**  WORKER.ENAME||'의관리자는'||MANAGER.  ------------------------------------  BLAKE의 관리자는 KING이다  CLARK의 관리자는 KING이다  . . . . . . . . .  13 rows selected. |

**1.8 Set Operators**

하나 이상의 테이블로부터 자료를 검색하는 또 다른 방법은 SET연산자를 이용하는 방법이 있다. 즉 SET연산자를 이용하여 여러 개의 SELECT문장을 연결하여 작성할 수 있다.

**1.8.1) Syntax**

|  |
| --- |
| **SELECT** \* | column1[, column2, column3, . . . . ]  **FROM** table1  . . . . . . . . .  **SET** operator  **SELECT** \* | column1[, column2, column3, . . . . ]  **FROM** table2  . . . . . . . . .  **[ORDER BY** column | expression]; |

☞ Guidelines

1. 첫번째 SELECT 구문에서 기술된 열과 두번째 SELECT 구문에서 기술된 열들은 좌측부터 1대1 대응하며 그 개수와 타입이 일치해야 한다.
2. FROM절 뒤에 기술되는 테이블은 같을 수도 있고 다를 수도 있다.
3. 출력되는 HARDING을 첫번째 SELECT구문에서 기술된 열이 출력된다.
4. ORDER BY는 단 한번만 기술 가능하고 SELECT 구문의 마지막에 기술한다.
5. SELECT문장은 위에서 아래로 수행되고 이를 변경하고자 할 경우는 괄호를 사용한다.

**1.8.2) SET 연산자의 종류**

|  |  |
| --- | --- |
| 종 류 | 설 명 |
| UNION | 각 결과의 합(합집합:중복되는 값은 한번 출력) |
| UNION ALL | 각 결과의 합(합집합) |
| INTERSET | 각 결과의 중복되는 부분만 출력(교집합) |
| MINUS | 첫번째 결과에서 두번째 결과를 뺌(차집합) |

**1.8.3) UNION과 UNION ALL의 차이**

양쪽에서 검색된 결과를 모두 출력한다. 아래의 두개의 SELECT 문장을 참조하여라.

SQL> SELECT deptno SQL> SELECT deptno

2 FROM dept 2 FROM dept

3 UNION 3 UNION ALL

4 SELECT deptno 4 SELECT deptno

5 FROM emp; 5 FROM emp;

A집합 B집합

A집합 B집합

DEPTNO DEPTNO

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

10 10

20 20

30 30

40 40

+

10

A집합 B집합

30

. . . . . .

18 rows selected.

**1.8.4) INTERSECT 연산자**

양쪽에서 검색된 자료만 출력한다. 아래의 SELECT문장을 참조하여라.

SQL> SELECT deptno

2 FROM dept

A집합 B집합

3 INTERSECT

4 SELECT deptno

5 FROM emp;

DEPTNO

---------

10

20

30

**1.8.5) MINUS 연산자**

두번째 SELECT문장에서 검색되지 않았던 값을 첫번째 SELECT문장에서 출력한다. 즉 첫번째 SELECT문장에서 두번째 SELECT문장에의 값을 뺀것을 출력한다. 아래의 SELECT문장을 참조하여라.

SQL> **SELECT deptno**

A집합 B집합

2 **FROM dept**

3 **MINUS**

4 **SELECT deptno**

5 **FROM emp;**

DEPTNO

---------

40

**◈ 연 습 문 제 ◈**

1. EMP 테이블에서 모든 사원에 대한 이름,부서번호,부서명을 출력하는 SELECT 문장을 작성하여라.

2. EMP 테이블에서 NEW YORK에서 근무하고 있는 사원에 대하여 이름,업무,급여,부서명을 출력하는 SELECT 문장을 작성하여라.

3. EMP 테이블에서 보너스를 받는 사원에 대하여 이름,부서명,위치를 출력하는 SELECT 문장을 작성하여라.

4. EMP 테이블에서 이름 중 L자가 있는 사원에 대하여 이름,업무,부서명,위치를 출력하는 SELECT 문장을 작성하여라.

5. 아래의 결과를 출력하는 SELECT 문장을 작성하여라.(관리자가 없는 King을 포함하여 모든 사원을 출력)

|  |
| --- |
| Employee Emp# Manager Mgr#  ---------- --------- ---------- ---------  KING 7839  BLAKE 7698 KING 7839  CLARK 7782 KING 7839  . . . . . . . . . .  14 rows selected. |

6. EMP 테이블에서 그들의 관리자 보다 먼저 입사한 사원에 대하여 이름,입사일,관리자 이름, 관리자 입사일을 출력하는 SELECT 문장을 작성하여라.

7. EMP 테이블에서 사원의 급여와 사원의 급여 양만큼 “\*”를 출력하는 SELECT 문장을 작성하여라. 단 “\*”는 100을 의미한다.

|  |
| --- |
| Employee and their salary  -----------------------------------------------------------------  KING \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  BLAKE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  CLARK \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  JONES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  MARTIN \*\*\*\*\*\*\*\*\*\*\*\*  ALLEN \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  TURNER \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  . . . . . . . . . .  14 rows selected. |