### join

입력으로 받는 데이터의 필드와 서브 쿼리 결과 필드를 비교해 결합(join)합니다.

join은 스트림 쿼리를 지원하지 않습니다. 스트림 쿼리에 조인을 적용하려면 streamjoin 명령어를 사용하세요.

#### 문법

join [type={cross|full|inner|left|leftonly|right|rightonly}] KEY\_FIELD, ... [ SUBQUERY ]

필수 매개변수

**KEY\_FIELD, ...**

조인의 기준이 키 필드. 필드 구분자로 쉼표(,)를 사용합니다. type=cross일 때, 예외적으로 키 필드 목록을 사용하지 않습니다.

**[ SUBQUERY ]**

입력 데이터와 조인할 데이터를 출력하는 서브쿼리문을 대괄호 쌍([ ])으로 감싸서 입력

선택 매개변수

**type={cross|full|inner|left|leftonly|right|rightonly}**

조인 유형을 지정합니다(기본값: inner).

1. cross: 'Catersian product'라고도 합니다. 입력 데이터의 집합(M개 레코드)과 서브쿼리 결과 집합(N개)을 결합해 M x N 개의 레코드로 구성된 집합을 출력합니다. cross 조인을 사용할 때, KEY\_FIELD를 지정하지 않아야 합니다.
2. full: 키가 일치하는 레코드는 병합해서, 키가 일치하지 않는 레코드는 있는 그대로 출력합니다. 데이터의 합집합과 유사합니다.
3. 키가 일치하면, 입력 데이터 레코드에 서브쿼리 필드를 결합해 출력합니다.
4. 키가 일치하지 않으면, 입력 데이터와 서브쿼리 데이터를 각각 그대로 출력합니다.
5. inner: 키가 일치하는 레코드만 결합해서 출력합니다. 키를 포함하지 않는 레코드는 출력하지 않습니다. 데이터의 교집합과 유사합니다. 일반적으로 조인이라 하면 "inner 조인"을 의미합니다.
6. left: 키가 일치하는 레코드는 결합해서 출력하고, 키가 일치하지 않는 레코드는 입력 데이터의 레코드만 출력합니다.
7. leftonly: 입력 레코드 집합 중에서 서브쿼리 결과 집합과 키가 일치하지 않는 레코드만 출력합니다. 키가 일치하는 레코드는 출력하지 않습니다.
8. right: 키가 일치하는 레코드는 결합해서 출력하고, 키가 일치하지 않는 레코드는 서브쿼리 결과만 출력합니다.
9. rightonly: 서브쿼리 결과 집합 중에서 입력 레코드 집합과 키가 일치하지 않는 레코드만 출력합니다. 키가 일치하는 레코드는 출력하지 않습니다.

#### 사용 예

**code** 필드를 키로 하여 inner 조인

# code 필드를 갖는 json을 입력 데이터로 전달 | json "[{'code':1}, {'code':2}, {'code':3}]" | # code, name 필드를 갖는 json을 반환하는 서브쿼리문 | # 입력 데이터와 서브쿼리문의 결과 데이터를 code 필드를 키로 하여 inner join | join code [ json "[ {'code':1, 'name':'foo'}, {'code':2, 'name':'bar'} ]" ]

**쿼리 실행 결과 (inner join)**

|  |  |
| --- | --- |
| code | name |
| 1 | foo |
| 2 | bar |

서브 쿼리에서 조회되는 결과를 제외하고 출력(leftonly 조인)

json "[ {'field1': 'A'}, {'field1': 'B'}, {'field1': 'C'}, {'field1': 'D'}]" | join type=leftonly field1 [ json "[ {'field1': 'A', 'field2': 'Foo'}, {'field1': 'D', 'field2': 'Bar'} ]" ]

**쿼리 실행 결과 (leftonly join)**

|  |
| --- |
| field1 |
| B |
| C |

위에서 실행한 쿼리문은 다음 쿼리문과 실행 결과가 동일합니다(left 조인의 응용).

json "[ {'field1': 'A'}, {'field1': 'B'}, {'field1': 'C'}, {'field1': 'D'}]" | join type=left field1 [ json "[ {'field1': 'A', 'field2': 'Foo'}, {'field1': 'D', 'field2': 'Bar'} ]" ] | search isnull(field2)

부서별 통계 출력, 통계 값이 존재하지 않더라도 모든 부서를 출력(right 조인)

json "[{'id': 1, '건수': 1000}, {'id':2, '건수': 2000}]" | join type=right id [ json "[ {'id':1, '부서':'영업'}, {'id':2, '부서':'운영'}, {'id':3, '부서':'기술'} ]" ]

**쿼리 실행 결과 (right join)**

|  |  |  |
| --- | --- | --- |
| id | 부서 | 건수 |
| 1 | 영업 | 1000 |
| 2 | 운영 | 2000 |
| 3 | 기술 |  |

문서보안과 매체제어 위반 로그를 계정 기준으로 조합하고, 일치하지 않는 경우 각 로그 출력 (full 조인)

json "[ {'계정':'bob', '문서보안위반': 1}, {'계정':'alice', '문서보안위반': 5} ]" | join type=full 계정 [ json "[ {'계정':'alice', '매체제어위반': 8}, {'계정':'clark', '매체제어위반': 3} ]" ]

**쿼리 실행 결과 (full join)**

|  |  |  |
| --- | --- | --- |
| 계정 | 문서보안위반 | 매체제어위반 |
| bob | 1 |  |
| alice | 5 | 8 |
| clark |  | 3 |

#### 호환성

type 중에서 cross, rightonly는 ENT-3.0.2003.0 버전부터 사용 가능합니다.