2.8 조인 (227~, 19)

8.1 조인 표현식

- 두개의 데이터셋에서 하나 이상의 키 값을 비교하고 결합 여부를 결정하는 표현식
- 조인 표현식의 평가 결과에 따라 두 개의 데이터셋을 조인

8.2 조인 타입

- 조인 타입
  - inner join
  - [left | right] outer join
  - left semi join, left anti join
  - natural join
  - cross join, cartesian join
- 그외 입력시 아래와 같은 오류 발생

```
java.lang.IllegalArgumentException: Unsupported join type 'INne␣R'.
Supported join types include: 'inner', 'outer', 'full', 'fullouter',
'leftouter', 'left', 'rightouter', 'right', 'leftsemi', 'leftanti',
'cross'.
```

Bash

- 대소문자 구분하지 않음

8.3 내부 조인 ("inner")

- 두 DataFrame 모두에 키가 존재하지 않으면 결과 DataFrame에서 볼 수 없다.
- joinType을 명시하지 않은 경우, 기본 joinType

```
val joinExpression = person.col("graduate_program") ===
graduateProgram("id") person.join( graduateProgram, joinExpression
).show() // person.join( graduateProgram, joinExpression, "inner"
).show() // 위와 동일 // person.join( graduateProgram, joinExpression,
"inNEr" ).show() // 위와 동일 +---+----------------+---------------+-----
----------+---+-------+--------------------+-----------+ | id|
name|graduate_program| spark_status| id| degree| department| school| +-
--+----------------+---------------+---------------+---+-------+------
--------------+-----------+ | 0| Bill Chambers| 0| [100]|
0|Masters|School of Informa...|UC Berkeley| | 1| Matei Zaharia| 1|[500,
250, 100]| 1| Ph.D.| EECS|UC Berkeley| | 2|Michael Armbrust| 1| [250,
100]| 1| Ph.D.| EECS|UC Berkeley| +---+----------------+--------------
-+---------------+---+-------+--------------------+-----------+
```

Scala

8.4 외부 조인 ("outer")

- 왼쪽이나 오른쪽 DataFrame에 일치하는 로우가 없다면 해당 위치에 null을 삽입
- 왼쪽, 오른쪽 모두 null만 존재하는 row 가능
- List

```
val joinExpression = person.col("graduate_program") ===
graduateProgram("id") person.join( graduateProgram, joinExpression,
"outer" ).show() // person.join( graduateProgram, joinExpression,
"full" ).show() // 위와 동일 // person.join( graduateProgram,
joinExpression, "fullouter" ).show() // 위와 동일 +----+----------------+-
--------------+---------------+----+-------+--------------------+-----
------+ | id| name|graduate_program| spark_status| id| degree|
department| school| +----+----------------+---------------+-----------
----+----+-------+--------------------+-----------+ | 1| Matei Zaharia|
1|[500, 250, 100]| 1| Ph.D.| EECS|UC Berkeley| | 2|Michael Armbrust| 1|
[250, 100]| 1| Ph.D.| EECS|UC Berkeley| | 9| Kwangsun Noh| 9|[500, 250,
100]|null| null| null| null| |null| null| null| null| 2|Masters|
EECS|UC Berkeley| | 0| Bill Chambers| 0| [100]| 0|Masters|School of
Informa...|UC Berkeley| +----+----------------+---------------+-------
--------+----+-------+--------------------+-----------+
```

Scala

8.5 왼쪽 외부 조인 ("left_outer")

- 오른쪽 DataFrame에 일치하는 로우가 없다면 해당 위치에 null을 삽입
- 왼쪽 DataFrame 기준으로 Join
- 오른쪽 DataFrame에는 null만 존재하는 row 가능

```scala
val joinExpression = person.col("graduate_program") ===
graduateProgram("id") person.join( graduateProgram, joinExpression,
"left_outer" ).show() // person.join( graduateProgram, joinExpression,
"leftouter" ).show() // 위와 동일 // person.join( graduateProgram,
joinExpression, "left" ).show() // 위와 동일 +---+---------------+-------
--------+---------------+----+-------+--------------------+----------
+ | id| name|graduate_program| spark_status| id| degree| department|
school| +---+---------------+---------------+--------------+----+---
----+--------------------+----------+ | 0| Bill Chambers| 0| [100]|
0|Masters|School of Informa...|UC Berkeley| | 1| Matei Zaharia| 1|[500,
250, 100]| 1| Ph.D.| EECS|UC Berkeley| | 9| Kwangsun Noh| 9|[500, 250,
100]|null| null| null| null| | 2|Michael Armbrust| 1| [250, 100]| 1|
Ph.D.| EECS|UC Berkeley| +---+---------------+---------------+-------
--------+----+-------+--------------------+----------+
```

Scala

8.6 오른쪽 외부 조인 ("right_outer")

- 왼쪽 DataFrame에 일치하는 로우가 없다면 해당 위치에 null을 삽입
- 오른쪽 DataFrame 기준으로 Join
- 왼쪽 DataFrame에는 null만 존재하는 row 가능

```scala
val joinExpression = person.col("graduate_program") ===
graduateProgram("id") person.join( graduateProgram, joinExpression,
"right_outer" ).show() // person.join( graduateProgram, joinExpression,
"rightouter" ).show() // 위와 동일 // person.join( graduateProgram,
joinExpression, "right" ).show() // 위와 동일 +----+---------------+-----
----------+--------------+---+-------+--------------------+----------
-+ | id| name|graduate_program| spark_status| id| degree| department|
school| +----+---------------+---------------+--------------+---+---
----+--------------------+----------+ | 0| Bill Chambers| 0| [100]|
0|Masters|School of Informa...|UC Berkeley| |null| null| null| null|
```

```
2|Masters| EECS|UC Berkeley| | 2|Michael Armbrust| 1| [250, 100]| 1|
Ph.D.| EECS|UC Berkeley| | 1| Matei Zaharia| 1|[500, 250, 100]| 1|
Ph.D.| EECS|UC Berkeley| +---+----------------+----------------+------
--------+---+-------+--------------------+-----------+
```

Scala

8.7 왼쪽 세미 조인 ("left_semi")

- 오른쪽 DataFrame은 결과에 포함되지 않음. 값이 존재하는지 확인하는 용도
- SQL의 IN 필터와 유사
- "right_semi"는 존재하지 않음

```
val joinExpression = person.col("graduate_program") ===
graduateProgram("id") person.join( graduateProgram, joinExpression,
"left_semi" ).show() //person.join( graduateProgram, joinExpression,
"leftsemi" ).show() // 위와 동일 +---+----------------+----------------+--
-------------+ | id| name|graduate_program| spark_status| +---+--------
--------+----------------+---------------+ | 0| Bill Chambers| 0|
[100]| | 1| Matei Zaharia| 1|[500, 250, 100]| | 2|Michael Armbrust| 1|
[250, 100]| +---+----------------+----------------+---------------+
```

Scala

8.8 왼쪽 안티 조인 ("left_anti")

- 오른쪽 DataFrame에서 찾을 수 없는 키의 로우만 결과에 포함. 값이 존재하지 않는지 확인하는 용도
- SQL의 NOT IN 필터와 유사
- "right_anti"는 존재하지 않음

```
val joinExpression = person.col("graduate_program") ===
graduateProgram("id") person.join( graduateProgram, joinExpression,
"left_anti" ).show() //person.join( graduateProgram, joinExpression,
"leftanti" ).show() // 위와 동일 +---+------------+----------------+------
---------+ | id| name|graduate_program| spark_status| +---+------------
+----------------+---------------+ | 9|Kwangsun Noh| 9|[500, 250, 100]|
+---+------------+----------------+---------------+
```

Scala

8.9 자연 조인

- 두 DataFrame에 동일한 컬럼명이 존재할 때, 조인하는 컬럼을 암시적으로 추정
- inner join, [left | right | full] outer join 가능

```
graduateProgram.withColumnRenamed("id", "graduate_program")
.createOrReplaceTempView("renamedGraduateProgram") // inner
spark.sql(""" SELECT * FROM person NATURAL JOIN renamedGraduateProgram
""").show spark.sql(""" SELECT * FROM person NATURAL INNER JOIN
renamedGraduateProgram """).show // 위와 동일 +---------------+---+------
---------+--------------+-------+------------------+-----------+
|graduate_program| id| name| spark_status| degree| department| school|
+---------------+---+--------------+--------------+-------+-------
-----------+-----------+ | 0| 0| Bill Chambers| [100]|Masters|School
of Informa...|UC Berkeley| | 1| 1| Matei Zaharia|[500, 250, 100]|
Ph.D.| EECS|UC Berkeley| | 1| 2|Michael Armbrust| [250, 100]| Ph.D.|
EECS|UC Berkeley| +---------------+---+--------------+--------------
-+-------+------------------+-----------+ // full outer spark.sql("""
SELECT * FROM person NATURAL FULL OUTER JOIN renamedGraduateProgram
""").show spark.sql(""" SELECT * FROM person NATURAL FULL JOIN
renamedGraduateProgram """).show // 위와 동일 +---------------+----+-----
----------+--------------+-------+------------------+-----------+
|graduate_program| id| name| spark_status| degree| department| school|
+---------------+----+--------------+--------------+-------+-------
------------+-----------+ | 1| 1| Matei Zaharia|[500, 250, 100]|
Ph.D.| EECS|UC Berkeley| | 1| 2|Michael Armbrust| [250, 100]| Ph.D.|
EECS|UC Berkeley| | 9| 9| Kwangsun Noh|[500, 250, 100]| null| null|
null| | 2|null| null| null|Masters| EECS|UC Berkeley| | 0| 0| Bill
Chambers| [100]|Masters|School of Informa...|UC Berkeley| +-----------
----+----+--------------+--------------+-------+------------------
+-----------+ // left outer spark.sql(""" SELECT * FROM person NATURAL
LEFT OUTER JOIN renamedGraduateProgram """).show spark.sql(""" SELECT *
FROM person NATURAL LEFT JOIN renamedGraduateProgram """).show // 위와 동
일 +---------------+---+--------------+--------------+-------+------
-------------+-----------+ |graduate_program| id| name| spark_status|
degree| department| school| +---------------+---+--------------+----
----------+-------+------------------+-----------+ | 0| 0| Bill
Chambers| [100]|Masters|School of Informa...|UC Berkeley| | 1| 1| Matei
Zaharia|[500, 250, 100]| Ph.D.| EECS|UC Berkeley| | 9| 9| Kwangsun Noh|
[500, 250, 100]| null| null| null| | 1| 2|Michael Armbrust| [250, 100]|
Ph.D.| EECS|UC Berkeley| +---------------+---+--------------+-------
--------+-------+------------------+-----------+ // right outer
```

```
spark.sql(""" SELECT * FROM person NATURAL RIGHT OUTER JOIN
renamedGraduateProgram """).show spark.sql(""" SELECT * FROM person
NATURAL RIGHT JOIN renamedGraduateProgram """).show // 위와 동일 +--------
--------+----+---------------+---------------+-------+---------------
----+-----------+ |graduate_program| id| name| spark_status| degree|
department| school| +---------------+----+---------------+-----------
----+-------+-------------------+-----------+ | 0| 0| Bill Chambers|
[100]|Masters|School of Informa...|UC Berkeley| | 2|null| null|
null|Masters| EECS|UC Berkeley| | 1| 2|Michael Armbrust| [250, 100]|
Ph.D.| EECS|UC Berkeley| | 1| 1| Matei Zaharia|[500, 250, 100]| Ph.D.|
EECS|UC Berkeley| +---------------+----+---------------+------------
--+-------+-------------------+-----------+
```

Scala

8.10 교차 조인 (카테시안 조인)

- 왼쪽 DataFrame의 모든 로우를 오른쪽 DataFrame의 모든 로우와 결합
- 왼쪽 DataFrame 로우 x 오른쪽 DataFrame 로우 갯수만큼의 결과

ex1) joinExpression이 있는 경우, cross 조인이 원하는 형태로 안될 수 있다.

```
val joinExpression = person.col("graduate_program") ===
person.col("graduate_program") person.join( graduateProgram,
joinExpression, "cross" ).show() +---+---------------+--------------
+--------------+---+-------+-------------------+-----------+ | id|
name|graduate_program| spark_status| id| degree| department| school| +-
--+---------------+---------------+--------------+---+-------+------
--------------+-----------+ | 0| Bill Chambers| 0| [100]|
0|Masters|School of Informa...|UC Berkeley| | 1| Matei Zaharia| 1|[500,
250, 100]| 1| Ph.D.| EECS|UC Berkeley| | 2|Michael Armbrust| 1| [250,
100]| 1| Ph.D.| EECS|UC Berkeley| +---+---------------+--------------
-+--------------+---+-------+-------------------+-----------+
```

Scala

ex2) joinExpression을 항상 참이 되도록하면, cross 조인을 원하는 형태로 수행되게 할 수 있다.

```
val joinExpression = lit(true) person.join( graduateProgram,
joinExpression, "cross" ).show() person.crossJoin( graduateProgram
).show() // 위와 동일 +---+---------------+--------------+------------
```

```
--+---+-------+------------------+----------+ | id|
name|graduate_program| spark_status| id| degree| department| school| +-
--+---------------+----------------+--------------+---+-------+------
-------------+----------+ | 0| Bill Chambers| 0| [100]|
0|Masters|School of Informa...|UC Berkeley| | 0| Bill Chambers| 0|
[100]| 2|Masters| EECS|UC Berkeley| | 0| Bill Chambers| 0| [100]| 1|
Ph.D.| EECS|UC Berkeley| | 1| Matei Zaharia| 1|[500, 250, 100]|
0|Masters|School of Informa...|UC Berkeley| | 1| Matei Zaharia| 1|[500,
250, 100]| 2|Masters| EECS|UC Berkeley| | 1| Matei Zaharia| 1|[500,
250, 100]| 1| Ph.D.| EECS|UC Berkeley| | 9| Kwangsun Noh| 9|[500, 250,
100]| 0|Masters|School of Informa...|UC Berkeley| | 9| Kwangsun Noh| 9|
[500, 250, 100]| 2|Masters| EECS|UC Berkeley| | 9| Kwangsun Noh| 9|
[500, 250, 100]| 1| Ph.D.| EECS|UC Berkeley| | 2|Michael Armbrust| 1|
[250, 100]| 0|Masters|School of Informa...|UC Berkeley| | 2|Michael
Armbrust| 1| [250, 100]| 2|Masters| EECS|UC Berkeley| | 2|Michael
Armbrust| 1| [250, 100]| 1| Ph.D.| EECS|UC Berkeley| +---+------------
---+---------------+--------------+---+-------+------------------+-
----------+
```

Scala

ex3) SQL을 통해 명시적으로 메서드를 호출할 수 있다.

```
val joinExpression = person.col("graduate_program") ===
graduateProgram("id") spark.sql(""" SELECT * FROM graduateProgram CROSS
JOIN person """).show +---+-------+------------------+----------+---
+---------------+---------------+--------------+ | id| degree|
department| school| id| name|graduate_program| spark_status| +---+-----
--+------------------+----------+---+---------------+-------------
--+---------------+ | 0|Masters|School of Informa...|UC Berkeley| 0|
Bill Chambers| 0| [100]| | 0|Masters|School of Informa...|UC Berkeley|
1| Matei Zaharia| 1|[500, 250, 100]| | 0|Masters|School of
Informa...|UC Berkeley| 9| Kwangsun Noh| 9|[500, 250, 100]| |
0|Masters|School of Informa...|UC Berkeley| 2|Michael Armbrust| 1|
[250, 100]| | 2|Masters| EECS|UC Berkeley| 0| Bill Chambers| 0| [100]|
| 2|Masters| EECS|UC Berkeley| 1| Matei Zaharia| 1|[500, 250, 100]| |
2|Masters| EECS|UC Berkeley| 9| Kwangsun Noh| 9|[500, 250, 100]| |
2|Masters| EECS|UC Berkeley| 2|Michael Armbrust| 1| [250, 100]| | 1|
Ph.D.| EECS|UC Berkeley| 0| Bill Chambers| 0| [100]| | 1| Ph.D.|
EECS|UC Berkeley| 1| Matei Zaharia| 1|[500, 250, 100]| | 1| Ph.D.|
EECS|UC Berkeley| 9| Kwangsun Noh| 9|[500, 250, 100]| | 1| Ph.D.|
EECS|UC Berkeley| 2|Michael Armbrust| 1| [250, 100]| +---+-------+-----
--------------+----------+---+---------------+--------------+-----
----------+
```

Scala

8.11.1 복합 데이터 타입의 조인

- 불리언을 반환하는 모든 표현식은 조인 표현식으로 간주할 수 있다.

ex) id가 중복되지 않게 수정 후, sparkStatus.id가 person.spark_status에 포함되는지를 기준으로 조인

```
import org.apache.spark.sql.functions.expr
person.withColumnRenamed("id", "personId") .join( sparkStatus,
expr("array_contains(spark_status, id)")) .show() +--------+-----------
------+---------------+---------------+---+-------------+ |personId|
name|graduate_program| spark_status| id| status| +--------+-----------
----+---------------+---------------+---+-------------+ | 0| Bill
Chambers| 0| [100]|100| Contributor| | 1| Matei Zaharia| 1|[500, 250,
100]|500|Vice President| | 1| Matei Zaharia| 1|[500, 250, 100]|250| PMC
Member| | 1| Matei Zaharia| 1|[500, 250, 100]|100| Contributor| | 9|
Kwangsun Noh| 9|[500, 250, 100]|500|Vice President| | 9| Kwangsun Noh|
9|[500, 250, 100]|250| PMC Member| | 9| Kwangsun Noh| 9|[500, 250,
100]|100| Contributor| | 2|Michael Armbrust| 1| [250, 100]|250| PMC
Member| | 2|Michael Armbrust| 1| [250, 100]|100| Contributor| +--------
+---------------+---------------+---------------+---+-------------+
```

Scala

8.11.2 중복 컬럼명 처리

ex1) 동일한 컬럼명이 존재하지만 오류 발생하지 않음

```
val gradProgramDupe = graduateProgram.withColumnRenamed("id",
"graduate_program") val joinExpr =
gradProgramDupe.col("graduate_program") ===
person.col("graduate_program") person.join(gradProgramDupe,
joinExpr).show() +---+---------------+---------------+--------------
+---------------+-------+-------------------+-----------+ | id|
name|graduate_program| spark_status|graduate_program| degree|
department| school| +---+---------------+---------------+-----------
---+---------------+-------+-------------------+-----------+ | 0|
```

```
Bill Chambers| 0| [100]| 0|Masters|School of Informa...|UC Berkeley| |
1| Matei Zaharia| 1|[500, 250, 100]| 1| Ph.D.| EECS|UC Berkeley| |
2|Michael Armbrust| 1| [250, 100]| 1| Ph.D.| EECS|UC Berkeley| +---+---
------------+---------------+--------------+---------------+-------
+------------------+----------+
```

Scala

ex2) select로 중복된 컬럼을 참조할 때 오류 발

```
val gradProgramDupe = graduateProgram.withColumnRenamed("id",
"graduate_program") val joinExpr =
gradProgramDupe.col("graduate_program") ===
person.col("graduate_program") person.join(gradProgramDupe,
joinExpr).select("graduate_program").show()
org.apache.spark.sql.AnalysisException: Reference 'graduate_program' is
ambiguous, could be: graduate_program#265, graduate_program#1317.; at
org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.resolve(Logical
Plan.scala:287) at
org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.resolveChildren
(LogicalPlan.scala:171) ...
```

Scala

해결 방법 1) 다른 조인 표현식 사용. 중복 컬럼 자동 제거

```
val gradProgramDupe = graduateProgram.withColumnRenamed("id",
"graduate_program") val joinExpr =
gradProgramDupe.col("graduate_program") ===
person.col("graduate_program") person.join(gradProgramDupe,
"graduate_program").show() +---------------+---+---------------+-----
----------+-------+------------------+----------+ |graduate_program|
id| name| spark_status| degree| department| school| +---------------+-
--+---------------+--------------+-------+------------------+------
-----+ | 0| 0| Bill Chambers| [100]|Masters|School of Informa...|UC
Berkeley| | 1| 1| Matei Zaharia|[500, 250, 100]| Ph.D.| EECS|UC
Berkeley| | 1| 2|Michael Armbrust| [250, 100]| Ph.D.| EECS|UC Berkeley|
+---------------+---+---------------+--------------+-------+--------
-----------+----------+
```

Scala

해결 방법 2) 조인 후 컬럼 제거

```
val gradProgramDupe = graduateProgram.withColumnRenamed("id",
"graduate_program") val joinExpr =
gradProgramDupe.col("graduate_program") ===
person.col("graduate_program") person.join(gradProgramDupe,
joinExpr).drop(person.col("graduate_program")).show() +---+------------
----+---------------+---------------+-------+------------------+----
-------+ | id| name| spark_status|graduate_program| degree| department|
school| +---+---------------+---------------+---------------+-------
+------------------+----------+ | 0| Bill Chambers| [100]|
0|Masters|School of Informa...|UC Berkeley| | 1| Matei Zaharia|[500,
250, 100]| 1| Ph.D.| EECS|UC Berkeley| | 2|Michael Armbrust| [250,
100]| 1| Ph.D.| EECS|UC Berkeley| +---+---------------+--------------
+---------------+-------+------------------+----------+
```

Scala

해결 방법 3) 조인 전 컬럼명 변경

```
val gradProgram3 = graduateProgram.withColumnRenamed("id", "grad_id")
```

Scala

8.12 스파크의 조인 수행 방식

- 노드간 네트워크 통신 전략, 노드별 연산 전략을 이해해야 함

8.12.1 네트워크 통신 전략

- 스파크는 조인 시 두 가지 클러스터 통신 방식 활용 (셔플 조인, 브로드캐스트 조인)
- 셔플 조인은 전체 노드 간 통신을 유발
- 시간이 흘러 CBO(cost-based optimizer)가 개선되고 더 나은 전략이 도입되는 경우 바뀔 수 있다.

case 1) 큰 테이블과 큰 테이블 조인

- 셔플 조인이 발생하게 되어, 전체 노드 간 통신이 발생

case 2) 큰 테이블과 작은 테이블 조인

- 테이블이 단일 워커 노드의 메모리 크기에 적합할 정도로 충분히 작은 경우 최적화 가능

  ⇒ 브로드캐스트 조인

  - 작은 DataFrame을 클러스터 전체 워커 노드에 복제
  - 최초 통신 후 전체 노드가 통신하는 현상 방지
- 스파크가 자동으로 브로드캐스트 조인으로 설정하기도 함
- 힌트를 줘서 브로드캐스트 조인으로 유인할 수 있으나 강제성은 없어서 무시될 수 있음
  - 힌트로 MAPJOIN, BROADCAST, BROADCASTJOIN 등을 설정 가능

```scala
import org.apache.spark.sql.functions.broadcast val joinExpr =
person.col("graduate_program") === graduateProgram.col("id")
person.join(broadcast(graduateProgram), joinExpr).explain() == Physical
Plan == *BroadcastHashJoin [graduate_program#265], [id#281], Inner,
BuildRight :- LocalTableScan [id#263, name#264, graduate_program#265,
spark_status#266] +- BroadcastExchange
HashedRelationBroadcastMode(List(cast(input[0, int, false] as bigint)))
+- LocalTableScan [id#281, degree#282, department#283, school#284]
import org.apache.spark.sql.functions.broadcast joinExpr:
org.apache.spark.sql.Column = (graduate_program = id)
```

Scala

case 3) 작은 테이블과 작은 테이블 조인

- 스파크가 조인 방식을 결정하도록 내버려 두는 것이 좋다.