SQL Joins: Comprehensive Review

DSC 208R - Data Management for Analytics

Introduction to SQL Joins

SQL joins are used to combine rows from two or more tables based on a related column between them.

Schemas Used in Examples

- **Movie** (name, year, genre)
- **ActedIN** (actorname, moviename)
- **Actor** (name, age)

Data for Examples

Movie Table:

Name	Year	Genre
Apocalypse Now	1979	War
The God Father	1972	Crime
Planet Earth II	2016	Nature Documentary

ActedIN Table:

Actorname	Moviename
Marlon Brando	Apocalypse Now
Al Pacino	The God Father
Marlon Brando	The God Father

Actor Table:

Name	Age
Marlon Brando	80
Al Pacino	82
De Niro	79

Types of SQL Joins

Inner Join

An **INNER JOIN** (or simply `JOIN`) returns only the rows that have matching values in both tables. Rows that do not have a match in the other table are excluded.

Query: What movies did each actor act in?

```
SELECT A.actorname, M.name
FROM ActedIN AS A JOIN Movie AS M
ON A.moviename = M.name;
```

Result:

actorname	name
Marlon Brando	Apocalypse Now
Al Pacino	The God Father
Marlon Brando	The God Father

Explanation: The query joins ‘ActedIN‘ and ‘Movie‘ tables where ‘moviename‘ in ‘ActedIN‘ matches ‘name‘ in ‘Movie‘. It then projects ‘actorname‘ and ‘name‘ (movie title) for the matched rows.

Left Outer Join

A **LEFT OUTER JOIN** (or `LEFT JOIN`) returns all rows from the left table, and the matching rows from the right table. If there is no match in the right table, 'NULL' values are returned for the right table's columns.

Query: Return all actors, and for each, the movies they acted in.

```
SELECT T1.name, T2.moviename
FROM Actor AS T1 LEFT OUTER JOIN ActedIN AS T2
ON T1.name = T2.actorname;
```

Result:

name	moviename
Marlon Brando	Apocalypse Now
Marlon Brando	The God Father
Al Pacino	The God Father
De Niro	NULL

Explanation: This query ensures all actors from the 'Actor' table (left table) are included in the result. If an actor has no corresponding entries in the 'ActedIN' table, 'NULL' is displayed for 'moviename'.

Right Outer Join

A **RIGHT OUTER JOIN** (or `RIGHT JOIN`) returns all rows from the right table, and the matching rows from the left table. If there is no match in the left table, 'NULL' values are returned for the left table's columns.

Query: Return all movies, and for each movie, the actors who acted in it.

```
SELECT T1.name, T2.actorname
FROM Movie AS T1 RIGHT OUTER JOIN ActedIN AS T2
ON T1.name = T2.moviename;
```

Result:

name	actorname
Apocalypse Now	Marlon Brando
The God Father	Al Pacino
The God Father	Marlon Brando
NULL	NULL

Explanation (clarified from original): The query intends to show all entries from the 'ActedIN' table (right table) and match them with 'Movie' names. However, based on the provided table data in, 'Planet Earth II' from the 'Movie' table is not present in the 'ActedIN.moviename' column. A correct 'RIGHT JOIN' of 'Movie' (T1) and 'ActedIN' (T2) *on* 'T1.name = T2.moviename' would include all 'ActedIN' entries (as shown) and *also* any 'Movie' entries that are not in 'ActedIN'. If 'ActedIN' has movie names not in 'Movie', they would appear with 'NULL' for 'Movie.name'. The provided example result focuses on the matched 'ActedIN' entries, which might omit 'NULL' rows for the Movie table if there are no 'ActedIN' records for those movies. For instance, if 'Planet Earth II' had no actors, it would *not* appear in this specific result, as 'ActedIN' is the right table. The example given in the PDF is actually incomplete for a full RIGHT JOIN illustration given the input data. A more typical result would show all rows from 'ActedIN' (the right table), along with matching movie details from 'Movie'. The sample output matches the 'ActedIN' table exactly.

Full Outer Join

A **FULL OUTER JOIN** (or `FULL JOIN`) returns all rows when there is a match in either the left (T1) or the right (T2) table. It includes:

- All matching rows.
- All rows from the left table that have no match in the right table (with 'NULL' for right table columns).
- All rows from the right table that have no match in the left table (with 'NULL' for left table columns).

Query:

```
SELECT *
FROM Actor FULL OUTER JOIN ActedIN
ON Actor.name = ActedIN.actorname;
```

Result:

name	age	actorname	moviename
Marlon Brando	80	Marlon Brando	Apocalypse Now
Marlon Brando	80	Marlon Brando	The God Father
Al Pacino	82	Al Pacino	The God Father
De Niro	79	NULL	NULL
NULL	NULL	NULL	NULL

Explanation (clarified from original): This query performs a full outer join between 'Actor' and 'ActedIN' tables on 'Actor.name = ActedIN.actorname'. The result includes:

- Matched rows (Marlon Brando, Al Pacino) with their age and acted movies.
- Rows from 'Actor' that have no match in 'ActedIN' (e.g., De Niro, who hasn't acted in listed movies), with 'NULL' for 'actorname' and 'moviename' from 'ActedIN'.
- *Theoretically*, rows from 'ActedIN' that have no match in 'Actor' (e.g., if there was an 'ActedIN' entry for an actor not in the 'Actor' table), with 'NULL' for 'Actor.name' and 'Actor.age'. The example result shows a 'NULL' row that might represent this, but based on the provided 'Actor' and 'ActedIN' data, all 'actorname' values in 'ActedIN' ('Marlon Brando', 'Al Pacino') *do* exist in 'Actor.name'. Thus, a 'NULL, NULL, NULL, NULL' row isn't directly derivable from the given data, suggesting either an omitted example or an implicit case.

Union in SQL

The **UNION** operator combines the result sets of two or more `SELECT` statements into a single result set.

- Each `SELECT` statement within the 'UNION' must have the same number of columns.
- The columns must have compatible data types.
- The columns must be in the same order.
- 'UNION' by default removes duplicate rows. To include duplicates, use 'UNION ALL'.

Query: Return all names from 'Actor' and 'Movie' tables

```
SELECT name FROM Actor
UNION
SELECT name FROM Movie;
```

Result:

name
Al Pacino
Apocalypse Now
De Niro
Marlon Brando
Planet Earth II
The God Father

Explanation: This query combines the 'name' column from the 'Actor' table and the 'name' column from the 'Movie' table. 'UNION' ensures that only distinct names appear in the final result, regardless of which table they originated from.