Joining Data in SQL

BA770 Lab Session

Questrom School of Business, Boston University

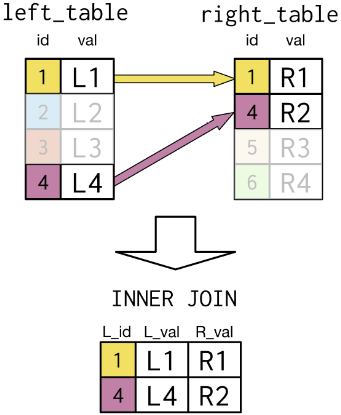
August 5, 2019

Topics

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN
- CROSS JOIN

- UNION
- UNION ALL
- INTERSECT
- EXCEPT
- Semi-join, Anti-join

INNER JOIN



left_table

id	val
1	L1
2	L2
3	L3
4	L4

right_table

id	val
1	R1
4	R2
5	R3
6	R4

INNER JOIN

L_id	L_val	R_val
1	L1	R1
4	L4	R2

Figure: INNER JOIN

INNER JOIN

- **SELECT** columns
  FROM table1
  INNER JOIN table2
  ON table1.column1 = table2.column2
- **SELECT** columns
  FROM table1
  INNER JOIN table2
  USING (column)

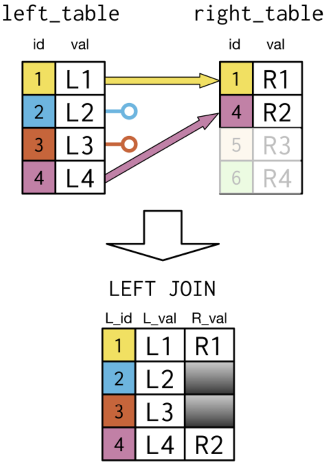
- Use **USING** keyword when the key field is in the same name in both tables. Remember to wrap the key field with parentheses.

INNER JOIN - self-join

- **SELECT** columns
  FROM table **AS** a
  INNER JOIN table **AS** b
  ON a.column1 = b.column2

- Often used for constructing combinations.
- Use table alias.
- You may include multiple conditions in **ON** clause combined with **AND**, **OR**, etc.

LEFT JOIN



left_table

id	val
1	L1
2	L2
3	L3
4	L4

right_table

id	val
1	R1
4	R2
5	R3
6	R4

LEFT JOIN

L_id	L_val	R_val
1	L1	R1
2	L2	
3	L3	
4	L4	R2

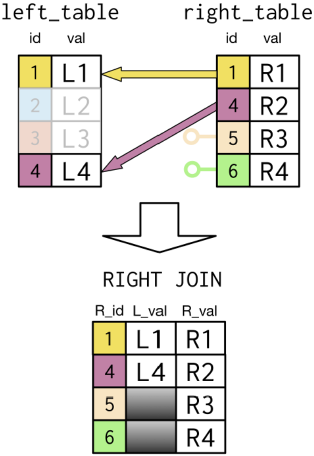
Figure: LEFT JOIN

LEFT JOIN

- **SELECT** columns
  FROM table1
  LEFT JOIN table2
  ON table1.column1 = table2.column2
- **SELECT** columns
  FROM table1
  LEFT JOIN table2
  USING (column)

- **LEFT JOIN** marks the values as missing in the right table for those that don't have a match.
- Duplicate rows may appear after **LEFT JOIN** if the key field in the left table have multiple matches in the right table.
- You may use multiple **LEFT JOIN**s in one query. Be careful about the order of tables.

RIGHT JOIN



left_table

id	val
1	L1
2	L2
3	L3
4	L4

right_table

id	val
1	R1
4	R2
5	R3
6	R4

RIGHT JOIN

R_id	L_val	R_val
1	L1	R1
4	L4	R2
5		R3
6		R4

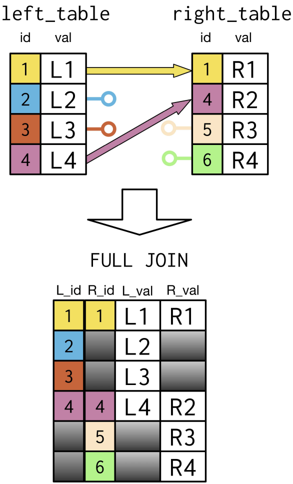
Figure: RIGHT JOIN

RIGHT JOIN

- **SELECT** columns
  FROM table1
  RIGHT JOIN table2
  ON table1.column1 = table2.column2
- **SELECT** columns
  FROM table1
  RIGHT JOIN table2
  USING (column)

- **RIGHT JOIN**s aren't as common as **LEFT JOIN**s, because you can always write a **RIGHT JOIN** as a **RIGHT JOIN**.

FULL JOIN



left_table

id	val
1	L1
2	L2
3	L3
4	L4

right_table

id	val
1	R1
4	R2
5	R3
6	R4

FULL JOIN

L_id	R_id	L_val	R_val
1	1	L1	R1
2		L2	
3		L3	
4	4	L4	R2
	5		R3
	6		R4

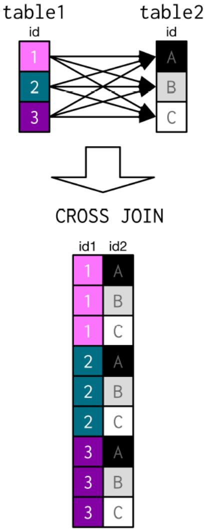
Figure: FULL JOIN

FULL JOIN

- **SELECT** columns
  FROM table1
  FULL JOIN table2
  ON table1.column1 = table2.column2
- **SELECT** columns
  FROM table1
  FULL JOIN table2
  USING (column)

- A **FULL JOIN** combines a **LEFT JOIN** and a **RIGHT JOIN**, bringing all records from both left and right tables and keep track of the missing values from both tables.
- **LEFT JOIN**, **RIGHT JOIN** and **FULL JOIN** are three types of **OUTER JOIN**s.

CROSS JOIN



id1	id2
1	A
1	B
1	C
2	A
2	B
2	C
3	A
3	B
3	C

Figure: CROSS JOIN

CROSS JOIN

- **SELECT** columns
  FROM table1
  CROSS JOIN table2
- A **CROSS JOIN** does not use **ON** or **USING**.

UNION

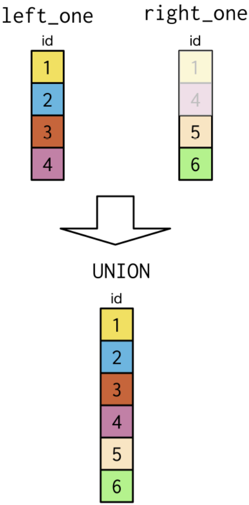
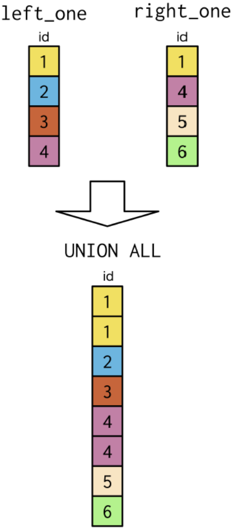


Figure: UNION

UNION

- **SELECT** column1 (**AS** new_col_name)
  FROM table1
  UNION
  SELECT column2
  FROM table2
- No double count of common values.
- Fields included in the operation must be of the same type.
- Often followed by an **ORDER BY** clause to sort the results and improve readability.
- Feel free to select multiple columns.
- BigQuery does not support **UNION**. Use **UNION DISTINCT** instead.

UNION ALL



left_one

id
1
2
3
4

right_one

id
1
4
5
6

UNION ALL

id
1
1
2
3
4
4
5
6

Figure: UNION ALL

UNION ALL

- **SELECT** column1 (**AS** new_col_name)
  FROM table1
  UNION ALL
  SELECT column2
  FROM table2
- Replicate values appearing in both tables.
- Distinction between **UNION (ALL)** and **JOIN**:
  UNION and **UNION ALL** simply bind records on top of one another in two tables, whereas **JOIN** does look-ups based on key fields given.

INTERSECT

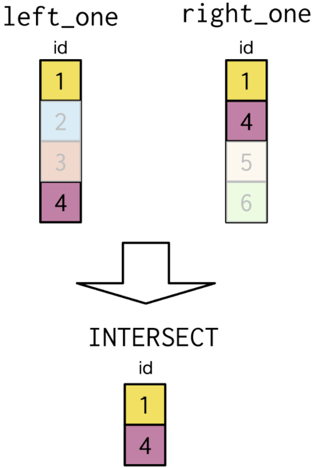


Figure: INTERSECT

INTERSECT

- **SELECT** column1 (**AS** new_col_name)
  FROM table1
  INTERSECT
  SELECT column2
  FROM table2
- When **INTERSECT** looks at multiple columns, it includes all columns in search.
- Distinction between **INTERSECT** and **JOIN**:
  INTERSECT looks for records in common, whereas **JOIN** matches individual key fields.

EXCEPT

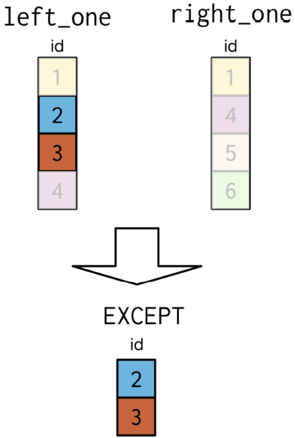


Figure: EXCEPT

EXCEPT

- **SELECT** column1 (**AS** new_col_name)
  FROM table1
  EXCEPT
  SELECT column2
  FROM table2

- Often followed by an **ORDER BY** clause to sort the results and improve readability.
- Feel free to select multiple columns.

Semi-join, Anti-join

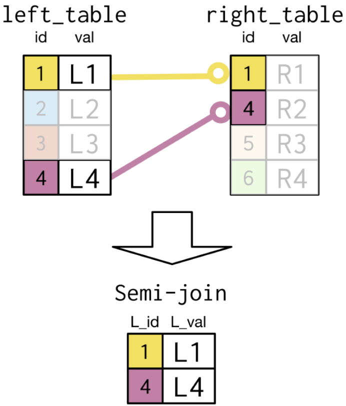


Figure: Semi-join

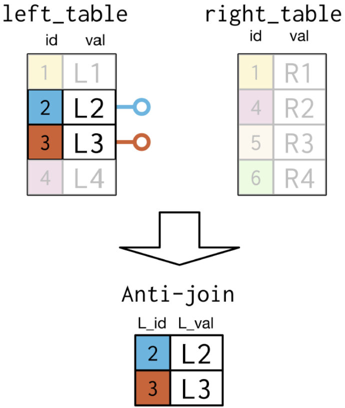


Figure: Anti-join

Semi-join, Anti-join

-
  ```
  SELECT column1
  FROM table1
  WHERE column2 (NOT) IN
     (SELECT column3
      FROM table2
      WHERE condition)
  ```

- Semi-join (anti-join) chooses records in the first table where a condition is (not) met in a second table.
- A semi-join/anti-join is involved with a subquery in **WHERE** clause.
- Often problems solved with these two types of joins can also be solved using **INNER JOIN**.