Joins

Paige Pfenninger and Kaitlyn Keil

Definition

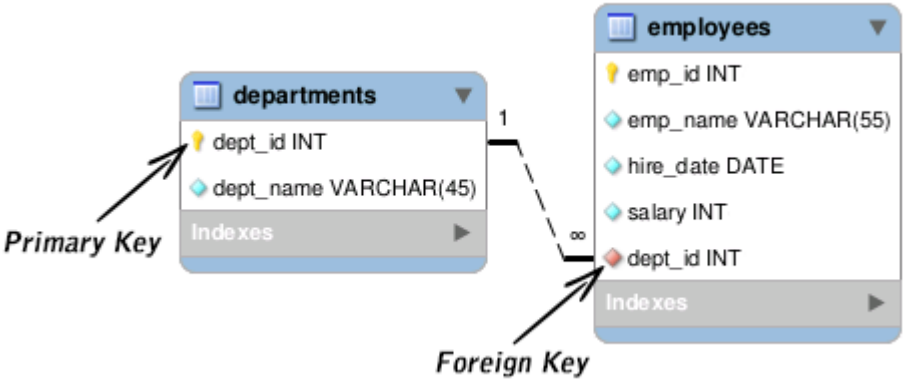
Relational database operation

Combine data tables based on some attribute, called the ‘join attribute’

> Similar to taking a product and selecting for conditions

Result in tuples that are the combination of the former tables

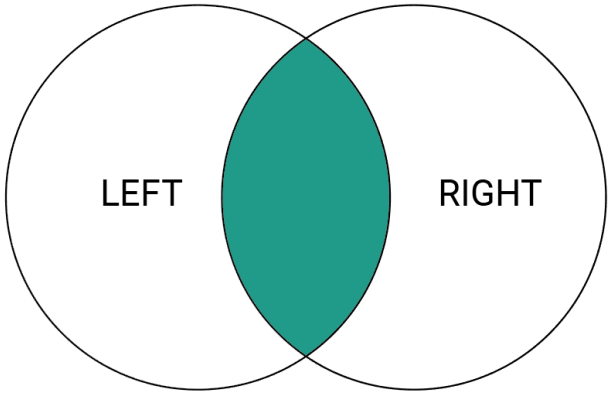
Often used to connect primary keys with foreign keys



https://www.tutorialrepublic.com/sql-tutorial/sql-constraints.php

Join Types

Inner Join



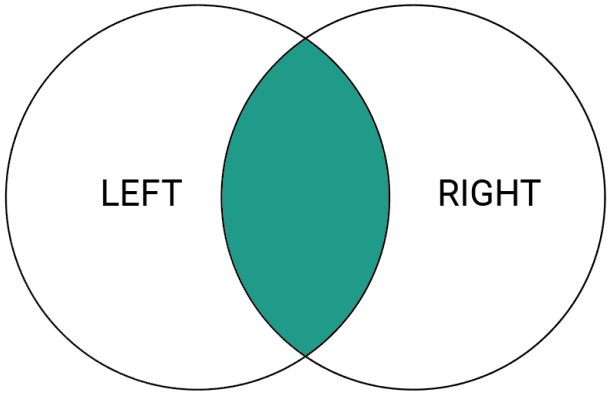
Occurs on some column name (or names)

Returns only the tuple combinations that satisfy the predicate

Naively, product and select on join predicate

```
SELECT *
FROM table1 INNER JOIN table2
  ON table1.column_name > table2.column_name;
```

Equi Join



Equi Joins are a subclass of inner joins.

Inner joins can use several different relational operators, whereas the relational operator in an equi join must be = (equal to)

Subcategory: Natural Joins

Automatically does Equi Join on a column with the same name in both relations

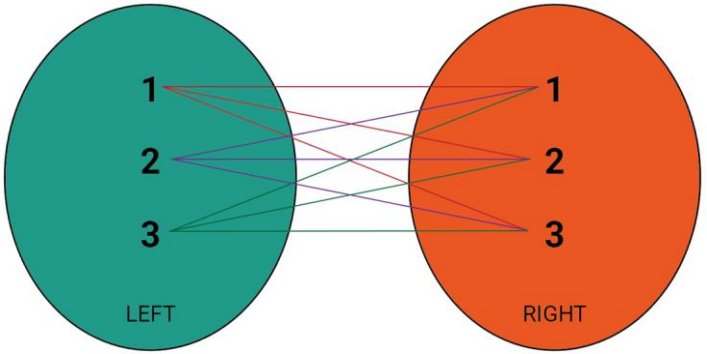
```
SELECT *
FROM table1 INNER JOIN table2
  ON table1.column_name = table2.column_name;
```

Cross Join

If no WHERE clause is specified, the cross join is just a Cartesian product

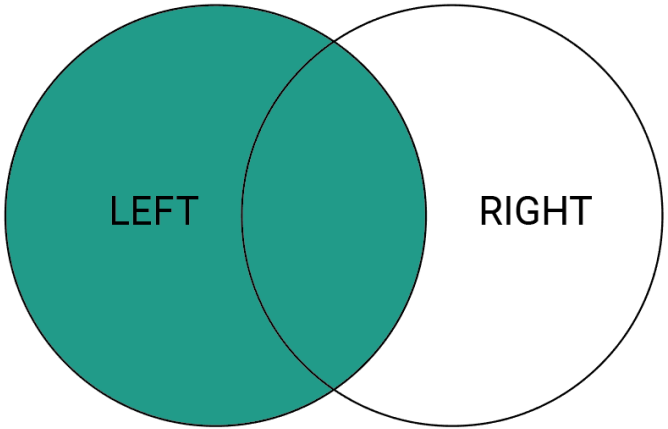
Otherwise, functions the same as an inner join

Useful when trying to find all possible combinations




```
SELECT *
FROM table1
CROSS JOIN table2;
```

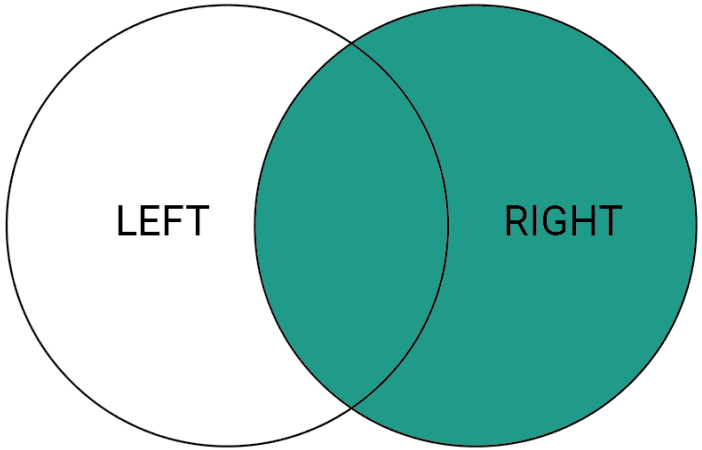
Left/Right Outer Join



Retains rows of the specified relation even if no match exists

If no match, populates the row with null values

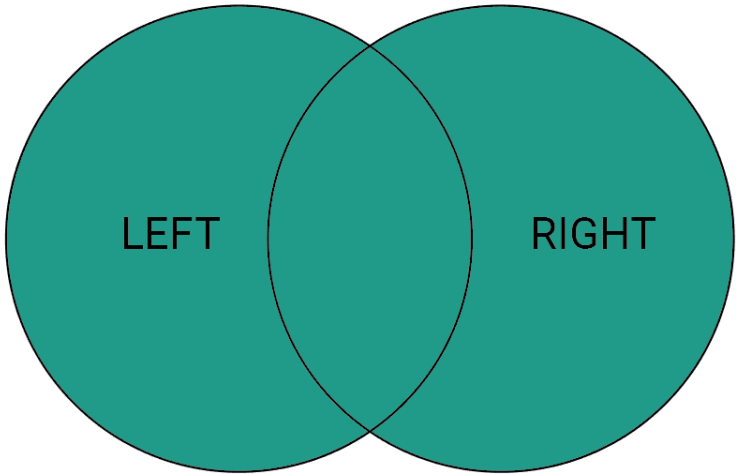
Has all the entries of the left/right, plus all values from inner join




```
SELECT *
FROM table1 LEFT JOIN table2
  ON table1.column_name = table2.column_name;
```

```
SELECT *
FROM table1 RIGHT JOIN table2
  ON table1.column_name = table2.column_name;
```

Full Outer Join



Retains all the entries of both relations, with predicate-satisfying tuples connected

If no match, populates the row with null values

No data loss

```
SELECT *
FROM table1 OUTER JOIN table2
  ON table1.column_name = table2.column_name;
```

Self Join

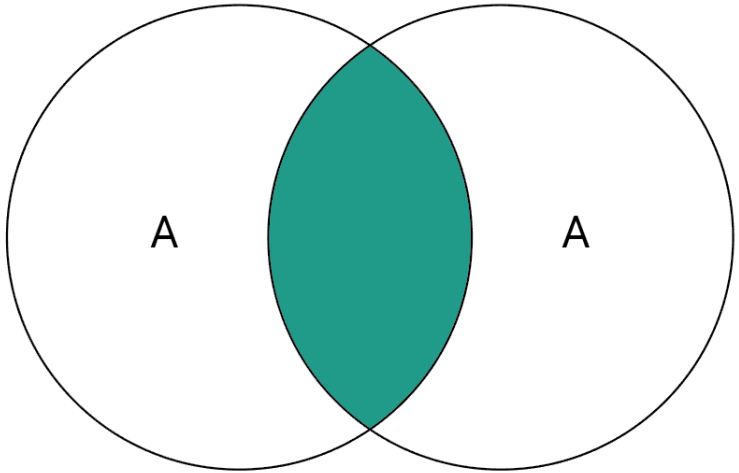
Joins a table to itself

To work properly, must examine two separate columns

> Consider EMPLOYEE with an emp_id and a emp_supv

Performs as if it has two copies of the relation without doubling it

Generally requires renaming




```
SELECT a.emp_id AS "Emp_ID",a.emp_name AS
"Employee Name",
b.emp_id AS "Supervisor ID",b.emp_name AS
"Supervisor Name"
FROM employee a, employee b
WHERE a.emp_supv = b.emp_id;
```

Reason

In advanced queries:

- Allows the user to specify scan order of relations
- Allows for filtering before joining

Generally more readable

NULL handling

NOT using it is basically just an implicit join

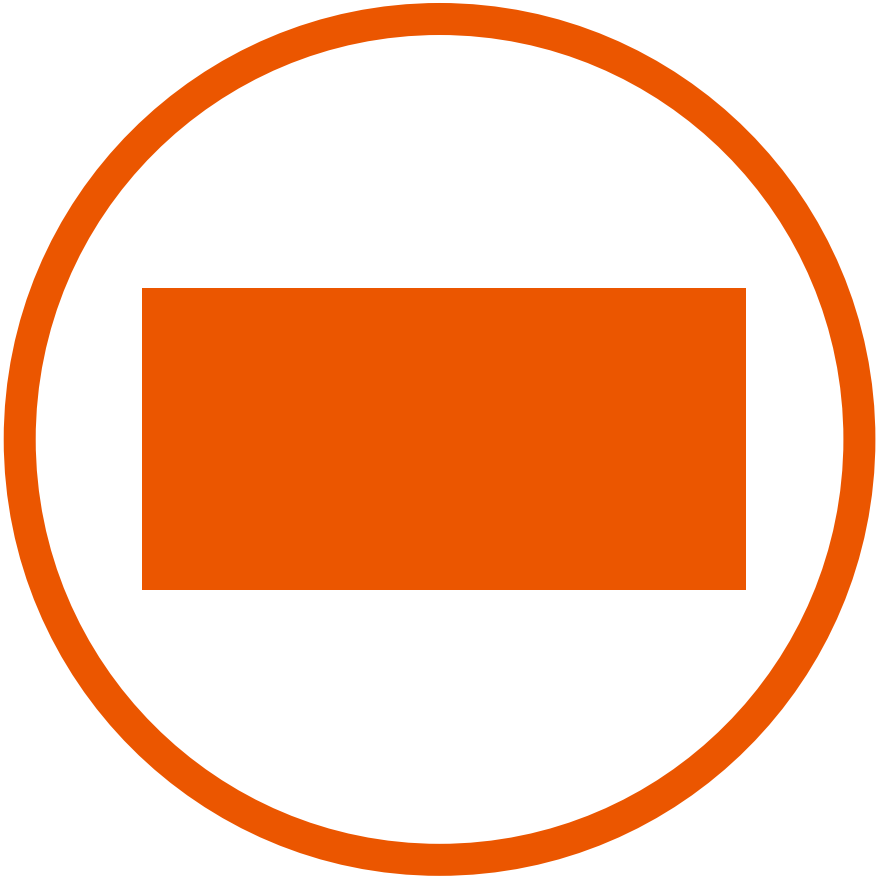
Issues

Can be very slow if not optimized in some way

Primary keys can be difficult in final result

- Multiple entries with same primary key
- NULL primary keys

Continues to be essentially product and select



Implementations

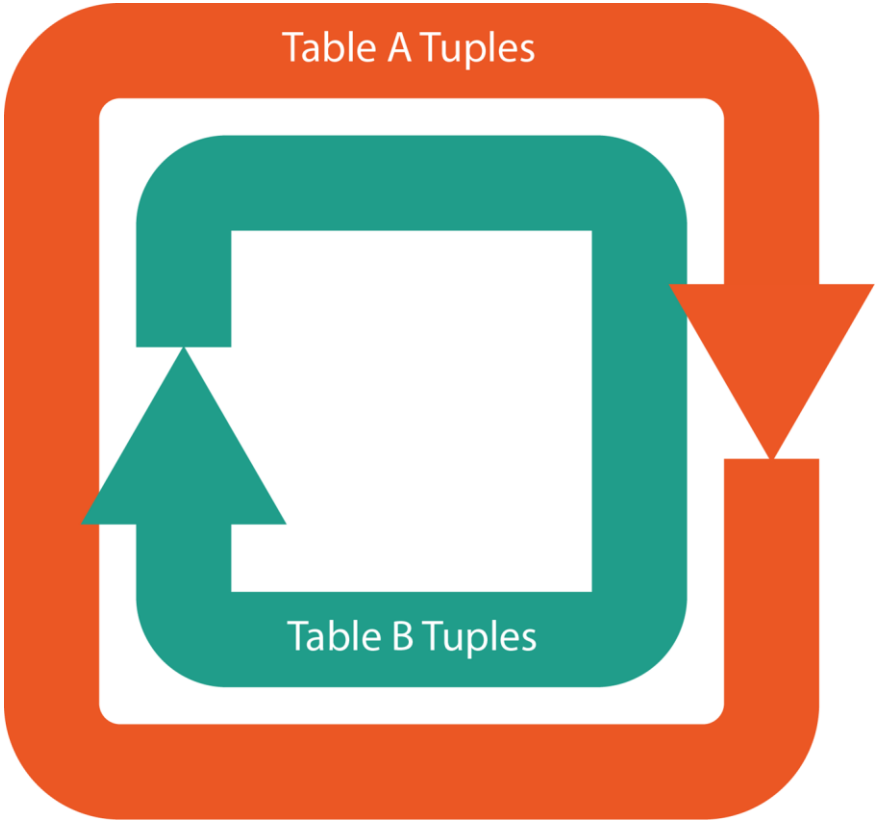
Nested Loop

For each tuple in Table A:

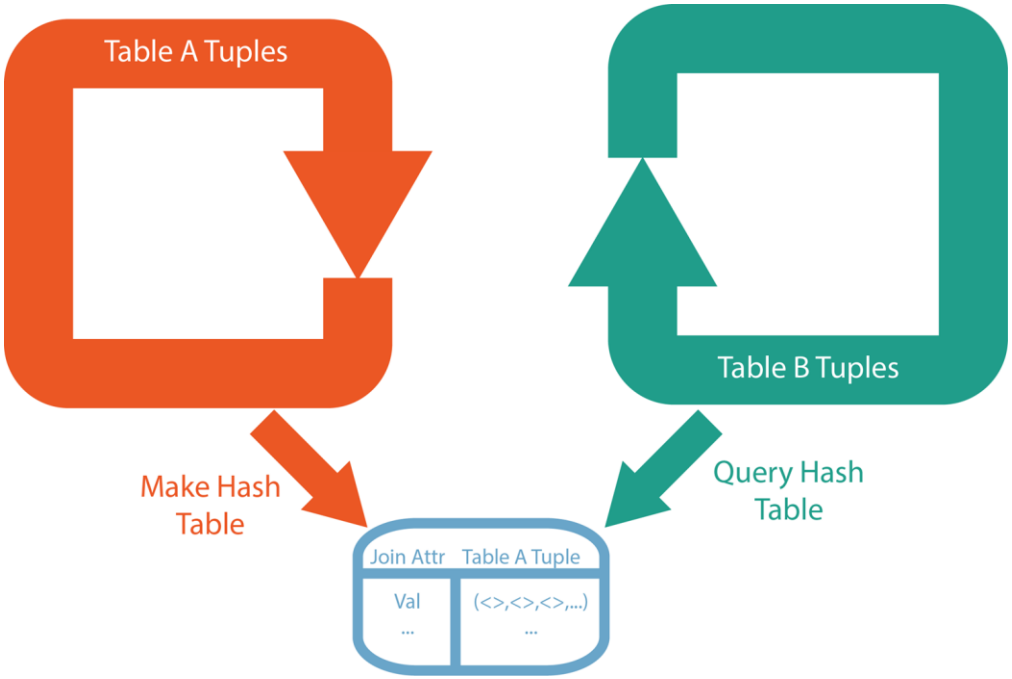
- Checks join attribute against each tuple in Table B
- Adds tuples which satisfy predicate to temporary table

Generally simple but slow

Works well for comparisons that aren't equality



Hash



For equality comparisons, nested loops might be too long

For hash joins:

- Iterate over first table to build temporary hash table
  - Key is join attribute
- Iterate over second table and probe for join attribute
- If found, add tuple to results

Speed at the cost of memory

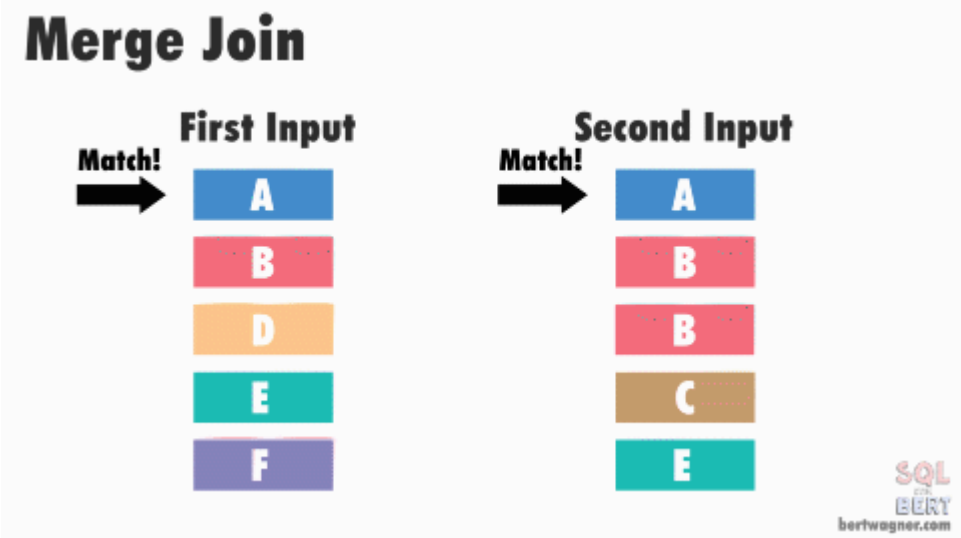
Merge

Faster than nested and less memory-intensive than hash

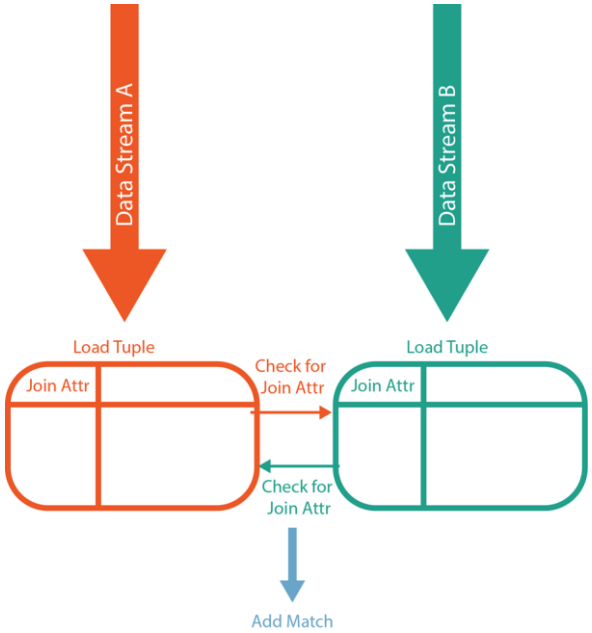
Sorts both tables on join attribute

Loops through both simultaneously, advancing when one value is behind

Good for comparing greater than/less than as well as equality



Symmetric Hash



Enhancement of Hash Join algorithm

Designed for data streams specifically

Creates a hash table from each side

Each new entry checks if there is an already existing one

If so, adds the new tuple

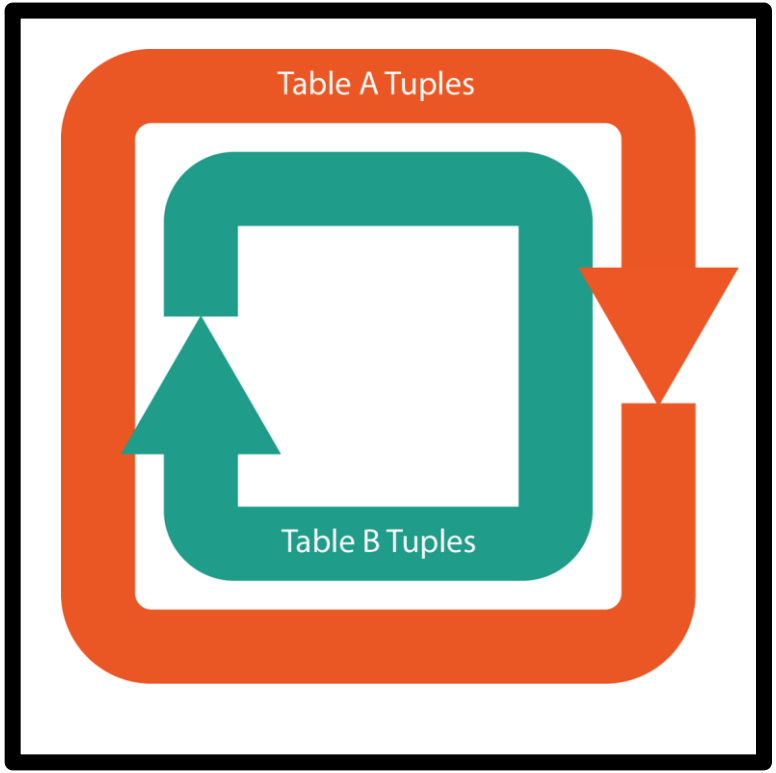
Block Nested Loop

As many blocks as possible in Table A are loaded into memory.

Table B is then scanned while looking for matches in Table A. Any matches found are put into result stream

Repeat process until all blocks in Table A have been loaded into memory

Generally only used for outer joins and semi joins



Questions

Works Cited

https://www.periscopedata.com/blog/how-joins-work

http://www.sql-join.com/

https://en.wikipedia.org/wiki/Join_(SQL)

https://mariadb.com/kb/en/library/block-based-join-algorithms/