Retrieving data from multiple tables

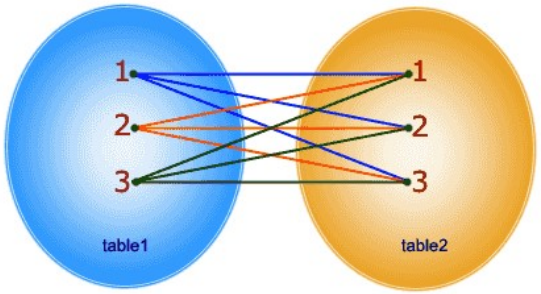
Displaying Data from Multiple Tables

Cartesian product

- In Mathematics, given two sets A and B, the Cartesian product of A x B is the set of all ordered pair (a,b), which a belongs to A and b belongs to B.
    - To create a Cartesian product of tables, you use the CROSS JOIN clause. The following illustrates the syntax of the CROSS JOIN clause:

      SELECT * FROM P1 CROSS JOIN P2;
    - When you perform a cross join of two tables, which have no relationship, we will get a Cartesian product of rows and columns of both tables.

SELECT * FROM table1 CROSS JOIN table2;

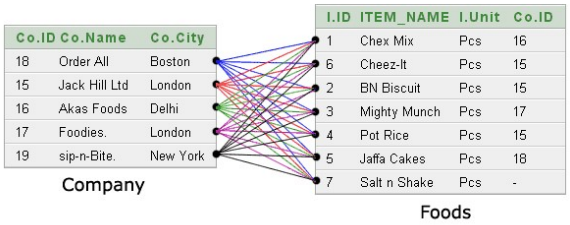


In CROSS JOIN, each row from 1st table joins with all the rows of another table.
If 1st table contain x rows and y rows in 2nd one the result set will be x * y rows.

Cartesian product

- A Cartesian product occurs when data is selected from two or more tables and no common relationship is specified in the WHERE clause.
    - If you do not specify a join condition for the tables listed in the FROM clause, database joins each row from the first table to every row in the second table.
    - If the first table has 3 rows and the second table has 4 rows, the result will have 12 rows.
    - If you add another table with 2 rows without specifying a join condition, the result will have 24 rows.
    - For the most part, Cartesian joins happen when there are many tables in the FROM clause and developers forget to include the join condition or they specify a wrong join condition.
- You should, therefore, avoid them. To avoid a Cartesian join, there should be at least n–1 join conditions when joining n tables. Sometimes you intentionally use Cartesian joins to generate large amounts of data, especially when testing applications

```
SELECT foods.item_name,foods.item_unit,
company.company_name,company.company_city
FROM foods
CROSS JOIN company;
```



Co.ID	Co.Name	Co.City
18	Order All	Boston
15	Jack Hill Ltd	London
16	Akas Foods	Delhi
17	Foodies.	London
19	sip-n-Bite.	New York

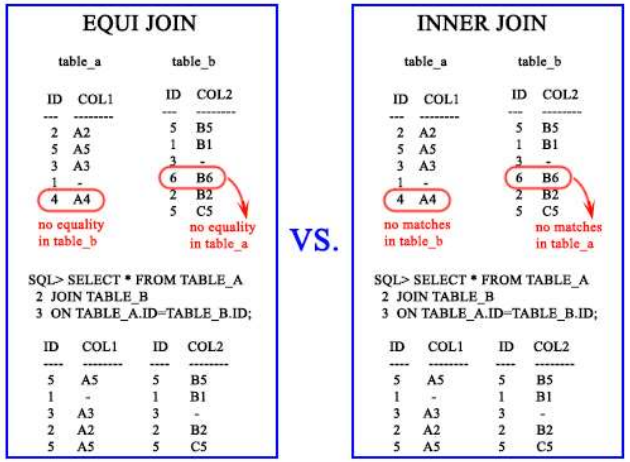
Company

I.ID	ITEM_NAME	I.Unit	Co.ID
1	Chex Mix	Pcs	16
6	Cheez-It	Pcs	15
2	BN Biscuit	Pcs	15
3	Mighty Munch	Pcs	17
4	Pot Rice	Pcs	15
5	Jaffa Cakes	Pcs	18
7	Salt n Shake	Pcs	-

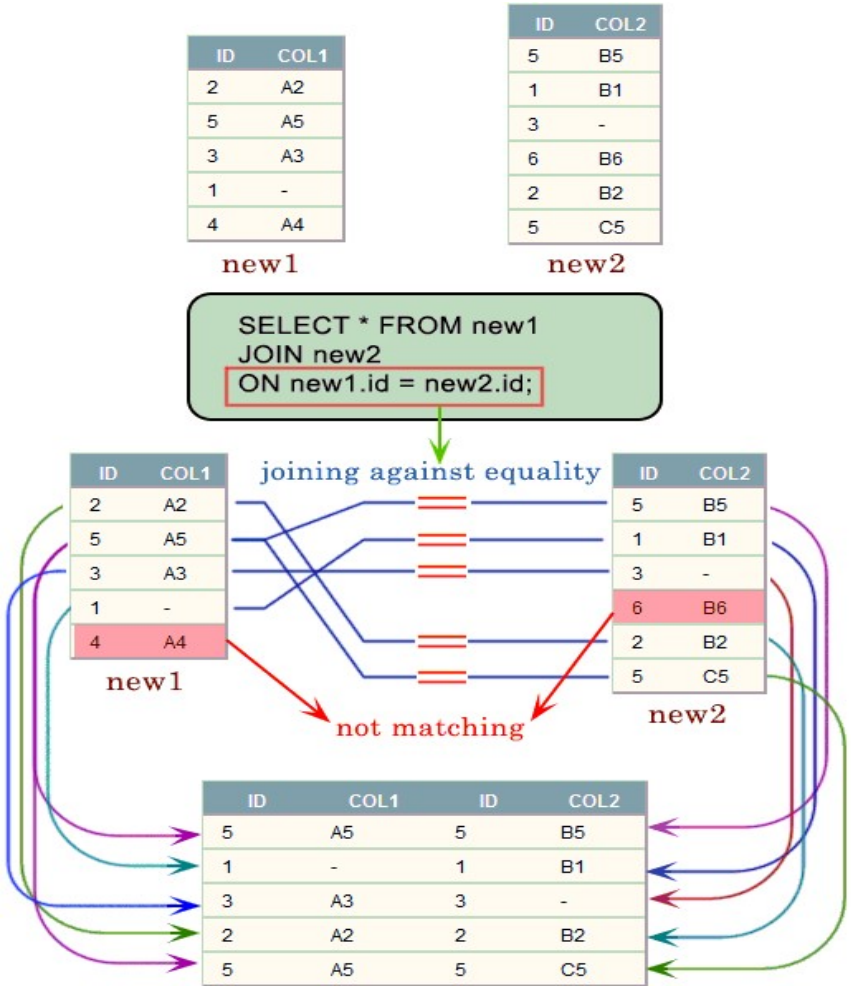
Foods

Matching using Equijoin

- Equi join returns the matching column values of the associated tables. It uses a comparison operator in the WHERE clause to refer equality.
    - Equijoin also can be performed by using JOIN keyword followed by ON keyword and then specifying names of the columns along with their associated tables to check equality.
    - An **equijoin** returns only the rows that have equivalent values for the specified columns.
    - An inner join is a join of two or more tables that returns only those rows (compared using a comparison operator) that satisfy the join condition.
- **What is the difference between Equi Join and Inner Join in SQL?**
    - An equijoin is a join with a join condition containing an equality operator. An equijoin returns only the rows that have equivalent values for the specified columns.
    - An inner join is a join of two or more tables that returns only those rows (compared using a comparison operator) that satisfy the join condition.



Matching using Equijoin



SQL JOINS – Non Equi JOIN

- Non Equi Join is a join condition containing something other than equality operator.
  - For example: sometimes you would like to join two tables that do not have a shared column, and seemingly have no join condition
  - Non-equi joins don’t have a specific keyword
  - They are defined by the type of operator in the join condition:
  - Anything but an equals sign means a non-equi join.
  - Below, we have some non-equi join operators and their meanings:

Operator	Meaning
“>”	Greater than
“>=”	Greater than or equal to
“<”	Less than
“<=”	Less than or equal to
“!=”	Not equal to
”<>”	Not equal to (ANSI Standard)

Matching using the EXISTS criteria

- EXISTS is only used to test if a subquery returns results. JOIN is used to extend a result set by combining it with additional fields from another table to which there is a relation.
  - EXISTS is used to return a boolean value, JOIN returns a whole other table.
- use EXISTS when:
  - You don't need to return data from the related table
  - You have duplicate in the related table (JOIN can cause duplicate rows if values are repeated)
  - You want to check existence (use instead of LEFT OUTER JOIN)

```
-- EXISTS
SELECT  *
FROM    tableA
WHERE   EXISTS (SELECT 1 FROM tableB WHERE title = 'Analyst' AND tableA.id = tableB.id);

-- JOIN (INNER is the default when only JOIN is specified)
SELECT  *
FROM    tableA
JOIN    tableB
   ON  tableA.id = tableB.id
WHERE   tableB.title = 'Analyst';
```

Retrieving data from multiple tables

- Questions