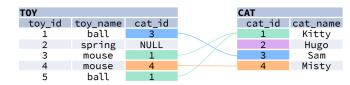


#### **JOINING TABLES**

JOIN combines data from two tables.



JOIN typically combines rows with equal values for the specified columns. Usually, one table contains a primary key, which is a column or columns that uniquely identify rows in the table (the cat\_id column in the cat table). The other table has a column or columns that **refer to the primary key columns** in the first table (the cat\_id column in the toy table). Such columns are foreign keys. The JOIN condition is the equality between the primary key columns in one table and columns referring to them in the other table.

#### JOIN

JOIN returns all rows that match the ON condition, JOIN is also called INNER JOIN.

SELECT *	toy_id	toy_name	cat_id	cat_id	cat_name
FROM toy	5	ball	1	1	Kitty
JOIN cat	3	mouse	1	1	Kitty
ON toy.cat_id = cat.cat_id;	1	ball	3	3	Sam
on coyreac_ia = cacreac_ia;	4	mouse	4	4	Misty

There is also another, older syntax, but it isn't recommended.

List joined tables in the FROM clause, and place the conditions in the WHERE clause.

SELECT \* FROM toy, cat WHERE toy.cat\_id = cat.cat\_id;

#### **JOIN CONDITIONS**

The JOIN condition doesn't have to be an equality – it can be any condition you want. JOIN doesn't interpret the JOIN condition, it only checks if the rows satisfy the given condition.

To refer to a column in the JOIN query, you have to use the full column name: first the table name, then a dot (.) and the

ON cat.cat\_id = toy.cat\_id

You can omit the table name and use just the column name if the name of the column is unique within all columns in the joined tables.

#### **NATURAL JOIN**

If the tables have columns with **the same name**, you can use

NATURAL JOIN instead of JOIN

SELECT \* FROM toy NATURAL JOIN cat;

The common column appears only once in the result table.

Note: NATURAL JOIN is rarely used in real life.

## **LEFT JOIN**

LEFT JOIN returns all rows from the left table with matching rows from the right table. Rows without a match are filled with NULLs. LEFT JOIN is also called LEFT OUTER JOIN.

```
SELECT *
                                         toy id toy name cat id cat id cat name
FROM toy
                                                    ball
LEFT JOIN cat
 ON toy.cat_id = cat.cat_id;
                                                    ball
                                                                               Sam
                                                                              Misty
                                                   mouse
                                                   spring
```

#### **RIGHT JOIN**

RIGHT JOIN returns all rows from the **right table** with matching rows from the left table. Rows without a match are filled with NULLs. RIGHT JOIN is also called RIGHT OUTER JOIN.

SELECT *
FROM toy
RIGHT JOIN cat
<pre>ON toy.cat_id = cat.cat_id;</pre>

toy_id	toy_name	cat_id	cat_id	cat_name
5	ball	1	1	Kitty
3	mouse	1	1	Kitty
NULL	NULL	NULL	2	Hugo
1	ball	3	3	Sam
4	mouse	4	4	Misty
			whole ri	ght table

#### **FULL JOIN**

FULL JOIN returns all rows from the left table and all rows from the right table. It fills the non-matching rows with NULLs. FULL JOIN is also called FULL OUTER JOIN.

FROM toy FULL JOIN cat ON toy.cat\_id = cat.cat\_id;

	toy_id	toy_name	cat_id	cat_id	cat_name
	5	ball	1	1	Kitty
	3	mouse	1	1	Kitty
	NULL	NULL	NULL	2	Hugo
	1	ball	3	3	Sam
	4	mouse	4	4	Misty
	2	spring	NULL	NULL	NULL
L		whole left table	whole ri	ght table	

### **CROSS JOIN**

CROSS JOIN returns all possible combinations of rows from the left and right tables.

FROM toy CROSS JOIN cat; Other syntax: SELECT \* FROM toy, cat;

toy_id	toy_name	cat_id	cat_id	cat_name
1	ball	3	1	Kitty
2	spring	NULL	1	Kitty
3	mouse	1	1	Kitty
4	mouse	4	1	Kitty
5	ball	1	1	Kitty
1	ball	3	2	Hugo
2	spring	NULL	2	Hugo
3	mouse	1	2	Hugo
4	mouse	4	2	Hugo
5	ball	1	2	Hugo
1	ball	3	3	Sam
• • •	• • •	• • •		

# **COLUMN AND TABLE ALIASES**

Aliases give a temporary name to a **table** or a **column** in a table.

AT AS C				OWNER AS	0
cat_id	cat_name	mom_id	owner_id	id	name
1	Kitty	5	1	1	John Smi
2	Hugo	1	2	2	Danielle Da
3	Sam	2	2		
4	Misty	1	NULL		

A column alias renames a column in the result. A table alias renames a table within the query. If you define a table alias,  $you\ must\ use\ it\ instead\ of\ the\ table\ name\ everywhere\ in\ the\ query.\ The\ AS\ keyword\ is\ optional\ in\ defining\ aliases.$ 

> o.name AS owner\_name, c.cat\_name FROM cat AS c JOIN owner AS o ON c.owner\_id = o.id;

cat_name	owner_name
Kitty	John Smith
Sam	Danielle Davis
Hugo	Danielle Davis

cat id tov id tov name cat name

ball

ball

mouse

Kitty

Sam

Misty

You can join a table to itself, for example, to show a parent-child relationship.

CAT AS C	hild			CAT AS mo	om		
cat_id	cat_name	owner_id	mom_id	cat_id	cat_name	owner_id	mom_id
1	Kitty	1	5	1	Kitty	1	5
2	Hugo	2	1	2	Hugo	2	1
3	Sam	2	2	3	Sam	2	2
4	Misty	NULL	1	4	Misty	NULL	1

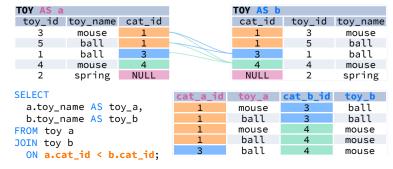
Each occurrence of the table must be given a different alias. Each column reference must be preceded with an

**SELECT** child.cat\_name AS child\_name, mom.cat\_name AS mom\_name FROM cat AS child JOIN cat AS mom ON child.mom\_id = mom.cat\_id;

child_name	mom_name
Hugo	Kitty
Sam	Hugo
Misty	Kitty
•	•

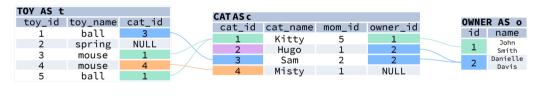
# **NON-EQUI SELF JOIN**

You can use a non-equality in the ON condition, for example, to show all different pairs of rows.



# **MULTIPLE JOINS**

You can join more than two tables together. First, two tables are joined, then the third table is joined to the result of the previous joining.





# JOIN WITH MULTIPLE CONDITIONS

You can use multiple JOIN conditions using the **ON** keyword once and the **AND** keywords as many times as you need.

CAT AS c					OWNE	R AS o	
cat_id	cat_name	mom_id	owner_id	age	id	name	ag
1	Kitty	5	1	17	1	John Smith	18
2	Hugo	1	2	10	2	Danielle Davis	10
3	Sam	2	2	5			
4	Misty	1	NULL	11			

# **SELECT**

cat\_name, o.name AS owner\_name, c.age AS cat\_age, o.age AS owner\_age FROM cat c JOIN owner o ON c.owner\_id = o.id

AND c.age < o.age;</pre>

cat_name	owner_name	age	age
Kitty	John Smith	17	18
Sam	Danielle Davis	5	10