SQL-JOINs Spickzettel

LearnSQL.de

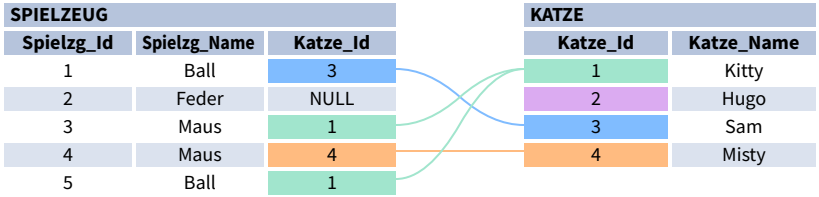
INHALTSVERZEICHNIS

SQL-JOINs Spickzettel

LearnSQL.de

TABELLEN VERBINDEN

`JOIN` kombiniert Daten aus zwei Tabellen.



SPIELZEUG

Spielzg_Id	Spielzg_Name	Katze_Id
1	Ball	3
2	Feder	NULL
3	Maus	1
4	Maus	4
5	Ball	1

KATZE

Katze_Id	Katze_Name
1	Kitty
2	Hugo
3	Sam
4	Misty

`JOIN` kombiniert typischerweise Zeilen mit gleichen Werten für die angegebenen Spalten. **Normalerweise** enthält eine Tabelle einen **Primärschlüssel**, d. h. eine oder mehrere Spalten, die Zeilen in der Tabelle eindeutig identifizieren (die Spalte `Katze_Id` in der Tabelle `Katze`). Die andere Tabelle hat eine oder mehrere Spalten, die **auf die Primärschlüsselspalten** der ersten Tabelle **verweisen** (die Spalte `Katze_Id` in der Tabelle `Spielzeug`). Solche Spalten sind **Fremdschlüssel**. Die `JOIN` Bedingung ist die Gleichheit zwischen den Primärschlüsselspalten in einer Tabelle und den Spalten, die sich auf sie in der anderen Tabelle beziehen.

SQL-JOINs Spickzettel

LearnSQL.de

JOIN

JOIN gibt alle Zeilen zurück, die mit der Bedingung ON übereinstimmen. JOIN wird auch als INNER JOIN bezeichnet.

```
SELECT *
FROM Spielzeug
JOIN Katze
  ON Spielzeug.Katze_Id = Katze.Katze_Id;
```

Spielzg_Id	Spielzg_Name	Katze_Id	Katze_Id	Katze_Name
5	Ball	1	1	Kitty
3	Maus	1	1	Kitty
1	Ball	3	3	Sam
4	Maus	4	4	Misty

Es gibt auch eine andere, ältere Syntax, die jedoch **nicht empfohlen** wird. Führen Sie die verbundenen Tabellen in der FROM -Klausel auf und stellen Sie die Bedingungen in die WHERE -Klausel.

```
SELECT *
FROM Spielzeug, Katze
WHERE Spielzeug.Katze_Id = Katze.Katze_Id;
```

SQL-JOINs Spickzettel

LearnSQL.de

JOIN-BEDINGUNGEN

Die Bedingung `JOIN` muss keine Gleichheit sein - sie kann jede beliebige Bedingung sein. `JOIN` interpretiert die Bedingung `JOIN` nicht, sondern prüft nur, ob die Zeilen die gegebene Bedingung erfüllen.
Um in der Abfrage `JOIN` auf eine Spalte zu verweisen, müssen Sie den vollständigen Spaltennamen verwenden: zuerst den Tabellennamen, dann einen Punkt (`.`) und den Spaltennamen:

```
  ON Katze.Katze_Id = Spielzeug.Katze_Id
```

Sie können den Tabellennamen weglassen und nur den Spaltennamen verwenden, wenn der Name der Spalte in allen Spalten der verbundenen Tabellen eindeutig ist. D.h. wenn er nur einmal vorkommt.

SQL-JOINs Spickzettel

LearnSQL.de

NATURAL JOIN

Wenn die Tabellen gleichnamige Spalten haben, können Sie `NATURAL JOIN` anstelle von `JOIN` verwenden.

```
SELECT *
FROM Spielzeug
NATURAL JOIN Katze;
```

Die gemeinsame Spalte erscheint nur einmal in der Ergebnistabelle.

Hinweis: `NATURAL JOIN` wird in der Praxis selten verwendet.

Katze_Id	Spielzg_Id	Spielzg_Name	Katze_Name
1	5	Ball	Kitty
1	3	Maus	Kitty
3	1	Ball	Sam
4	4	Maus	Misty

SQL-JOINs Spickzettel

LearnSQL.de

LEFT JOIN

`LEFT JOIN` gibt alle Zeilen aus der **linken Tabelle** mit übereinstimmenden Zeilen aus der rechten Tabelle zurück. Zeilen ohne Übereinstimmung werden mit `NULL`s aufgefüllt. `LEFT JOIN` wird auch `LEFT OUTER JOIN` genannt.

```
SELECT *
FROM Spielzeug
LEFT JOIN Katze
  ON Spielzeug.Katze_Id = Katze.Katze_Id;
```

Spielzg_Id	Spielzg_Name	Katze_Id	Katze_Id	Katze_Name
5	Ball	1	1	Kitty
3	Maus	1	1	Kitty
1	Ball	3	3	Sam
4	Maus	4	4	Misty
2	Feder	NULL	NULL	NULL

ganze linke Tabelle

SQL-JOINs Spickzettel

LearnSQL.de

RIGHT JOIN

`RIGHT JOIN` gibt alle Zeilen aus der **rechten Tabelle** mit übereinstimmenden Zeilen aus der linken Tabelle zurück. Zeilen ohne Übereinstimmung werden mit `NULL`s aufgefüllt. `RIGHT JOIN` wird auch `RIGHT OUTER JOIN` genannt.

```
SELECT *
FROM Spielzeug
RIGHT JOIN Katze
  ON Spielzeug.Katze_Id = Katze.Katze_Id;
```

Spielzg_Id	Spielzg_Name	Katze_Id	Katze_Id	Katze_Name
5	Ball	1	1	Kitty
3	Maus	1	1	Kitty
NULL	NULL	NULL	2	Hugo
1	Ball	3	3	Sam
4	Maus	4	4	Misty

ganze rechte Tabelle

SQL-JOINs Spickzettel

LearnSQL.de

FULL JOIN

`FULL JOIN` gibt alle Zeilen aus der **linken Tabelle** und alle Zeilen aus der **rechten Tabelle** zurück. Sie füllt die nicht übereinstimmenden Zeilen mit `NULL`s auf. `FULL JOIN` wird auch `FULL OUTER JOIN` genannt.

```
SELECT *
FROM Spielzeug
FULL JOIN Katze
  ON Spielzeug.Katze_Id = Katze.Katze_Id;
```

Spielzg_Id	Spielzg_Name	Katze_Id	Katze_Id	Katze_Name
5	Ball	1	1	Kitty
3	Maus	1	1	Kitty
NULL	NULL	NULL	2	Hugo
1	Ball	3	3	Sam
4	Maus	4	4	Misty
2	Feder	NULL	NULL	NULL
ganze linke Tabelle			ganze rechte Tabelle	

SQL-JOINs Spickzettel

LearnSQL.de

CROSS JOIN

`CROSS JOIN` gibt **alle möglichen Kombinationen** von Zeilen aus der linken und rechten Tabelle zurück.

```
SELECT *
FROM Spielzeug
CROSS JOIN Katze;
```

Other syntax:

```
SELECT *
FROM Spielzeug, Katze;
```

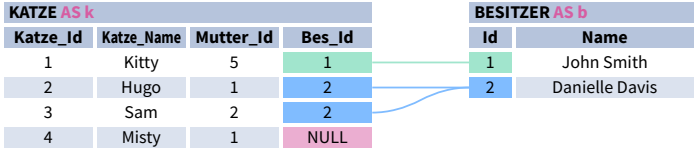
Spielzg_Id	Spielzg_Name	Katze_Id	Katze_Id	Katze_Name
1	Ball	3	1	Kitty
2	Feder	NULL	1	Kitty
3	Maus	1	1	Kitty
4	Maus	4	1	Kitty
5	Ball	1	1	Kitty
1	Ball	3	2	Hugo
2	Feder	NULL	2	Hugo
3	Maus	1	2	Hugo
4	Maus	4	2	Hugo
5	Ball	1	2	Hugo
1	Ball	3	3	Sam
...	...	...	...	...

SQL-JOINs Spickzettel

LearnSQL.de

SPALTEN- UND TABELLEN-ALIASE

Aliase geben einer **Tabelle** oder einer **Spalte** in einer Tabelle einen vorläufigen Namen.



KATZE AS k

Katze_Id	Katze_Name	Mutter_Id	Bes_Id
1	Kitty	5	1
2	Hugo	1	2
3	Sam	2	2
4	Misty	1	NULL

BESITZER AS b

Id	Name
1	John Smith
2	Danielle Davis

Ein **Spaltenalias** benennt eine Spalte im Ergebnis um. Ein **Tabellenalias** benennt eine Tabelle innerhalb der Abfrage um. Wenn Sie einen Tabellenalias definieren, müssen Sie ihn anstelle des Tabellennamens überall in der Abfrage verwenden. Das Schlüsselwort AS ist bei der Definition von Aliasen optional.

```
SELECT
  b.Name AS Bes_Name,
  k.Katze_Name
FROM Katze AS k
JOIN Besitzer AS b
  ON k.Bes_Id = b.Id;
```

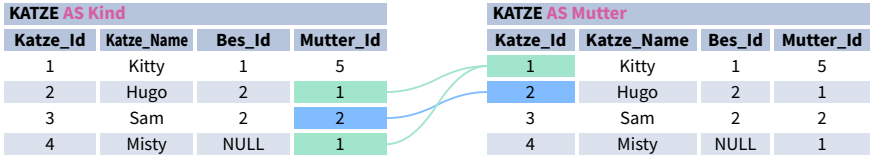
Katze_Name	Bes_Name
Kitty	John Smith
Sam	Danielle Davis
Hugo	Danielle Davis

SQL-JOINs Spickzettel

LearnSQL.de

SELF JOIN

Sie können eine Tabelle mit sich selbst verknüpfen, um z.B. eine Eltern-Kind-Beziehung darzustellen.



KATZE AS Kind

Katze_Id	Katze_Name	Bes_Id	Mutter_Id
1	Kitty	1	5
2	Hugo	2	1
3	Sam	2	2
4	Misty	NULL	1

KATZE AS Mutter

Katze_Id	Katze_Name	Bes_Id	Mutter_Id
1	Kitty	1	5
2	Hugo	2	1
3	Sam	2	2
4	Misty	NULL	1

Jedes Vorkommen der Tabelle muss mit einem **anderen Alias** versehen werden. Jeder Spaltenreferenz muss ein **entsprechender Tabellenalias** vorangestellt werden.

```
SELECT
  Kind.Katze_Name   AS Kind_Name,
  Mutter.Katze_Name AS Mutter_Name
FROM Katze AS Kind
JOIN Katze AS Mutter
  ON Kind.Mutter_Id = Mutter.Katze_Id;
```

Kind_Name	Mutter_Name
Hugo	Kitty
Sam	Hugo
Misty	Kitty

SQL-JOINs Spickzettel

LearnSQL.de

NON-EQUI SELF JOIN

Sie können z.B. eine **Ungleichheit** in der Bedingung ON verwenden, um **alle verschiedenen Zeilenpaare** anzuzeigen.

SPIELZEUG AS a

Spielzg_Id	Spielzg_Name	Katze_Id
3	Maus	1
5	Ball	1
1	Ball	3
4	Maus	4
2	Feder	NULL

SPIELZEUG AS b

Katze_Id	Spielzg_Id	Spielzg_Name
1	3	Maus
1	5	Ball
3	1	Ball
4	4	Maus
NULL	2	Feder

```
SELECT
  a.Spielzg_Name AS Spielzg_a,
  b.Spielzg_Name AS Spielzg_b
FROM Spielzeug a
JOIN Spielzeug b
  ON a.Katze_Id < b.Katze_Id;
```

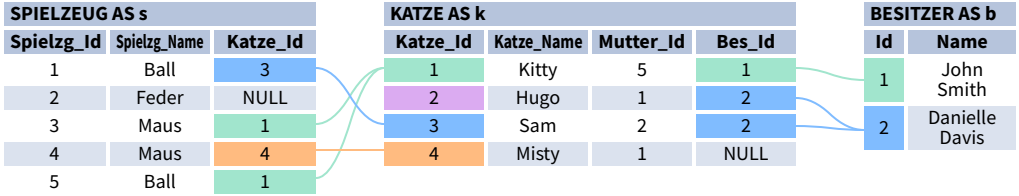
Katze_a_Id	Spielzg_a	Katze_b_Id	Spielzg_b
1	Maus	3	Ball
1	Ball	3	Ball
1	Maus	4	Maus
1	Ball	4	Maus
3	Ball	4	Maus

SQL-JOINs Spickzettel

LearnSQL.de

MEHRERE JOINS

Sie können mehr als zwei Tabellen miteinander verbinden. Zuerst werden zwei Tabellen verbunden, dann wird die dritte Tabelle mit dem Ergebnis der vorherigen Verbindung verbunden.



SPIELZEUG AS s

Spielzg_Id	Spielzg_Name	Katze_Id
1	Ball	3
2	Feder	NULL
3	Maus	1
4	Maus	4
5	Ball	1

KATZE AS k

Katze_Id	Katze_Name	Mutter_Id	Bes_Id
1	Kitty	5	1
2	Hugo	1	2
3	Sam	2	2
4	Misty	1	NULL

BESITZER AS b

Id	Name
1	John Smith
2	Danielle Davis

SQL-JOINs Spickzettel

LearnSQL.de

JOIN & JOIN

```
SELECT
  s.Spielzg_Name,
  k.Katze_Name,
  b.Name AS Bes_Name
FROM Spielzeug s
JOIN Katze k
  ON s.Katze_Id = k.Katze_Id
JOIN Besitzer b
  ON k.Bes_Id = b.Id;
```

Spielzg_Name	Katze_Name	Bes_Name
Ball	Kitty	John Smith
Maus	Kitty	John Smith
Ball	Sam	Danielle Davis

JOIN & LEFT JOIN

```
SELECT
  s.Spielzg_Name,
  k.Katze_Name,
  b.Name AS Bes_Name
FROM Spielzeug s
JOIN Katze k
  ON s.Katze_Id = k.Katze_Id
LEFT JOIN Besitzer b
  ON k.Bes_Id = b.Id;
```

Spielzg_Name	Katze_Name	Bes_Name
Ball	Kitty	John Smith
Maus	Kitty	John Smith
Ball	Sam	Danielle Davis
Maus	Misty	NULL

LEFT JOIN & LEFT JOIN

```
SELECT
  s.Spielzg_Name,
  k.Katze_Name,
  b.Name AS Bes_Name
FROM Spielzeug s
LEFT JOIN Katze k
  ON s.Katze_Id = k.Katze_Id
LEFT JOIN Besitzer b
  ON k.Bes_Id = b.Id;
```

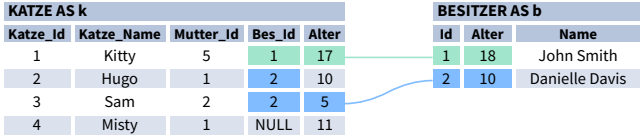
Spielzg_Name	Katze_Name	Bes_Name
Ball	Kitty	John Smith
Maus	Kitty	John Smith
Ball	Sam	Danielle Davis
Maus	Misty	NULL
Feder	NULL	NULL

SQL-JOINs Spickzettel

LearnSQL.de

JOIN MIT MEHREREN BEDINGUNGEN

Sie können mehrere `JOIN` Bedingungen verwenden, indem Sie das **ON** Schlüsselwort einmal und die **AND** Schlüsselwörter so oft wie nötig verwenden.



KATZE AS k

Katze_Id	Katze_Name	Mutter_Id	Bes_Id	Alter
1	Kitty	5	1	17
2	Hugo	1	2	10
3	Sam	2	2	5
4	Misty	1	NULL	11

BESITZER AS b

Id	Alter	Name
1	18	John Smith
2	10	Danielle Davis

```
SELECT
  Katze_Name,
  b.Name AS Bes_Name,
  k.Alter  AS Katze_Alter,
  b.Alter  AS Bes_Alter
FROM Katze k
JOIN Besitzer b
  ON k.Bes_Id = b.Id
 AND k.Alter < b.Alter;
```

Katze_Name	Bes_Name	Katze_Alter	Bes_Alter
Kitty	John Smith	17	18
Sam	Danielle Davis	5	10

Lerne alles auf LearnSQL.de