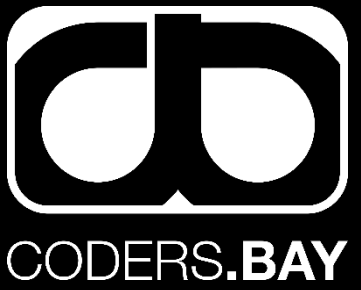
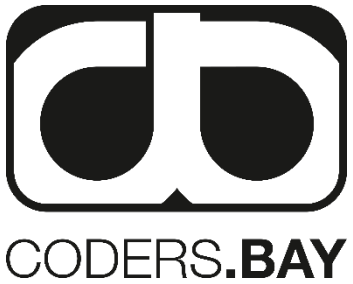
DIE WELT DER DATENBANKEN



WIEDERHOLUNG

Mysql Funktionen

MYSQL FUNKTIONEN

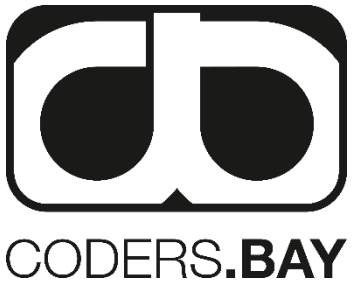


Spaltennamen der Zielrelation definieren mit AS

BSP:

```
SELECT salary AS Gehalt
FROM employees
```

MYSQL FUNKTIONEN

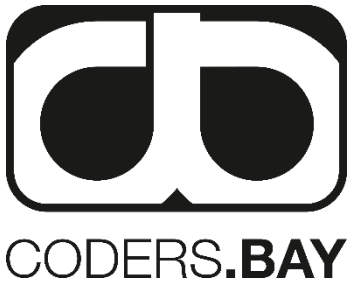


Durchschnitt mit `AVG()`

BSP:

```
SELECT AVG(salary) AS Durchschnittsgehalt
FROM employees
```

MYSQL FUNKTIONEN

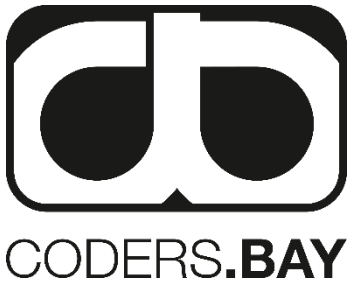


Summenbildung mit `SUM()`

BSP:

```
SELECT SUM(salary), job_id
FROM employees
GROUP BY job_id
ORDER BY SUM(salary) DESC;
```

MYSQL FUNKTIONEN



Zeichenketten zusammenführen mit `CONCAT()` / `CONCAT_WS()`

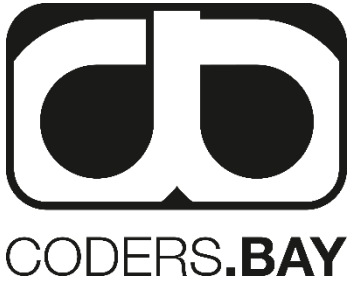
BSP:

```
SELECT CONCAT(first_name, ' ', last_name) AS Name
FROM employees

SELECT CONCAT_WS(' ', first_name, last_name) AS Name
FROM employees
```

WHERE-CLAUSE

WHERE-CLAUSE



Bei der Auswahl beziehungsweise Filterung mithilfe von

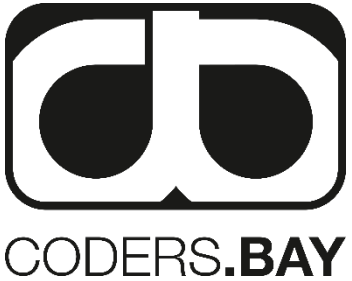
WHERE innerhalb der

SELECT-Anweisung kannst

du Vergleichsoperatoren anwenden

Operator	Bedeutung
=	gleich
<>	ungleich
>	größer als
>=	größer als oder gleich
<	kleiner als
<=	kleiner als oder gleich
NOT	Der Wahrheitswert einer Bedingung wird umgekehrt
AND	Alle Bedingungen müssen zutreffen
OR	Mindestens eine Bedingung muss zutreffen

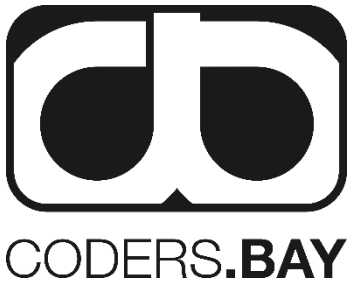
WHERE-CLAUSE



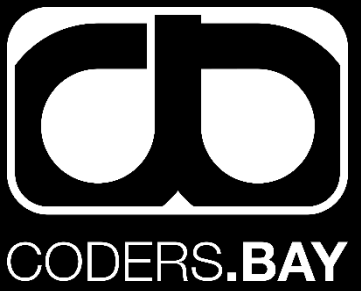
LIKE – für Zeichenketten

- **%** - Beliebige Anzahl unbekannter Zeichen
- _ = genau ein unbekanntes Zeichen

UNION

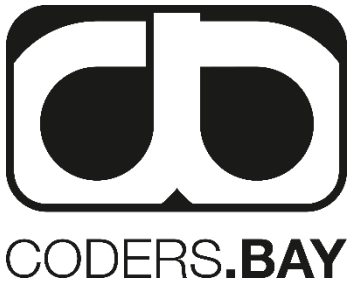



```
SELECT job_id, department_id
FROM employees
WHERE department_id = 10
UNION
SELECT job_id, department_id
FROM  employees
WHERE department_id = 20
```



JOINS

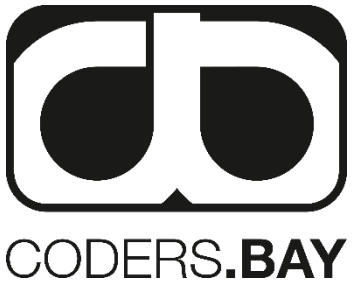
GRUNDLAGEN VON JOINS



JOIN Verbindet zwei oder mehr Relationen miteinander.

Abfrage von Daten über zwei oder mehr Tabellen

GRUNDLAGEN VON JOINS



CROSS JOIN - Jede Zeile von R1 verbunden mit jeder Zeile von R2

INNER JOIN - Verbindet alle Zeilen von R1 und R2 miteinander, wo ein Match gefunden wird.

OUTER JOIN - Verbindet alle Zeilen von R1 und R2 miteinander, wo ein Match gefunden wird. Wo keiner gefunden wird, wird der Rest mit NULL aufgefüllt.

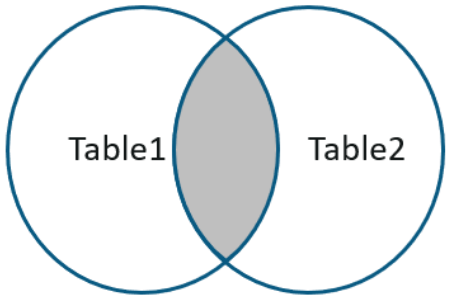
LEFT JOIN - Jede Zeile von R1 verbunden mit dazupassenden Zeilen von R2

RIGHT JOIN - Jede Zeile von R2 verbunden mit dazupassenden Zeilen von R1

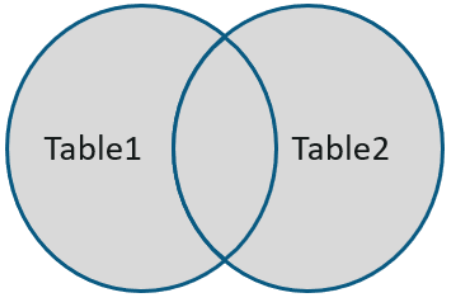
NATURAL JOIN - Natürlicher Verbund von R1 und R2 bei gleichnamiger Spalte

JOINS

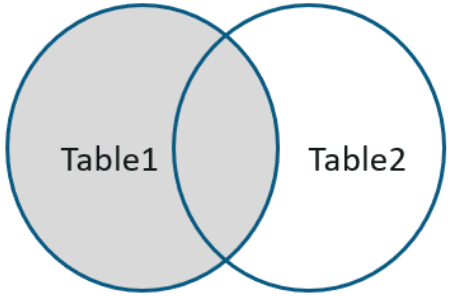
CODERS.BAY



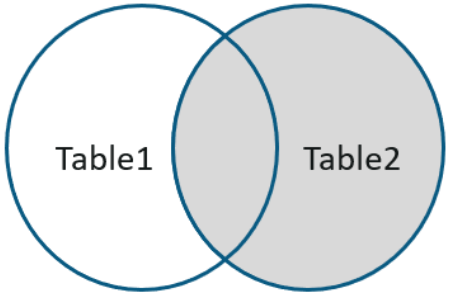
INNER JOIN



FULL JOIN



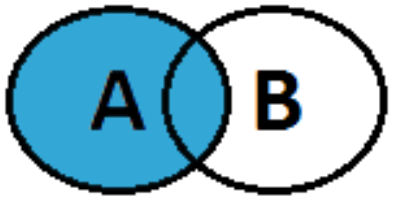
LEFT JOIN



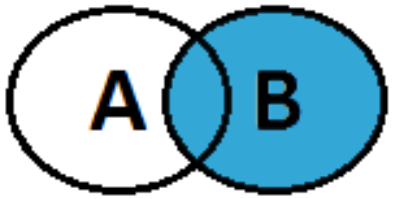
RIGHT JOIN

edureka!

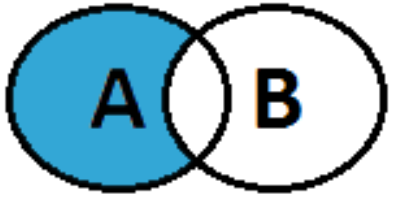
SQL JOIN Grundlagen
Die Welt der SQL JOINS



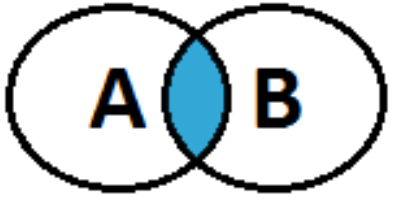

```
SELECT <Auswahl>
FROM TabelleA A
LEFT JOIN TabelleB B
ON A.Schlüssel = B.Schlüssel
```



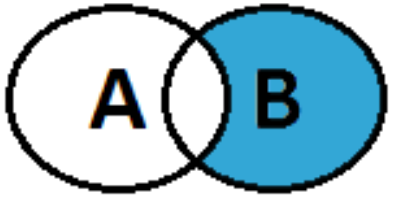

```
SELECT <Auswahl>
FROM TabelleA A
RIGHT JOIN TabelleB B
ON A.Schlüssel = B.Schlüssel
```



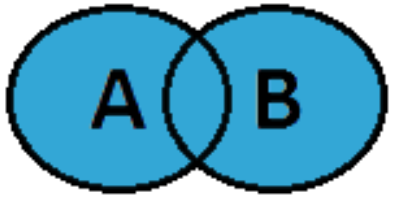

```
SELECT <Auswahl>
FROM TabelleA A
LEFT JOIN TabelleB B
ON A.Schlüssel = B.Schlüssel
WHERE B.Schlüssel IS NULL
```



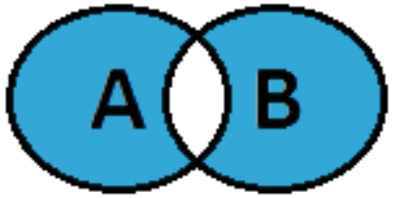

```
SELECT <Auswahl>
FROM TabelleA A
INNER JOIN TabelleB B
ON A.Schlüssel = B.Schlüssel
```




```
SELECT <Auswahl>
FROM TabelleA A
RIGHT JOIN TabelleB B
ON A.Schlüssel = B.Schlüssel
WHERE A.Schlüssel IS NULL
```




```
SELECT <Auswahl>
FROM TabelleA A
FULL OUTER JOIN TabelleB B
ON A.Schlüssel = B.Schlüssel
```




```
SELECT <Auswahl>
FROM TabelleA A
FULL OUTER JOIN Tabelle B
ON A.Schlüssel = B.Schlüssel
WHERE A.Schlüssel IS NULL
OR B.Schlüssel IS NULL
```


CODERS.BAY

GRUNDLAGEN VON JOINS

CODERS.BAY

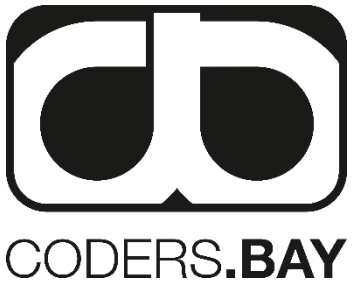
```
SELECT employees.last_name, departments.department_name
FROM employees
JOIN departments
ON employees.department_id = departments.department_id;
```

GRUNDLAGEN VON JOINS

CODERS.BAY

```
SELECT d.department_name, l.postal_code,
l.city, c.country_name
FROM departments d
JOIN locations l ON d.location_id = l.location_id
JOIN countries c ON l.country_id = c.country_id;
```

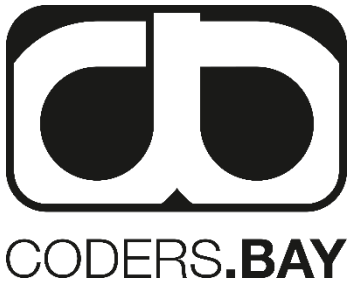
GRUNDLAGEN VON JOINS




```
SELECT e2.last_name AS Manager, e1.last_name AS Unterstellter
FROM employees e1
JOIN employees e2
ON e1.manager_id = e2.employee_id
ORDER BY Manager;
```

DATA DEFINITION LANGUAGE

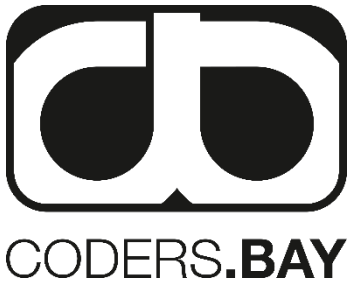
ERSTELLEN DER TABELLENSTRUKTUR



Erstellen mit `CREATE TABLE`

```
CREATE TABLE IF NOT EXISTS countries(
    country_id CHAR(2) NOT NULL,
    country_name VARCHAR(40),
    region_id INT,
    PRIMARY KEY(country_id)
    FOREIGN KEY(region_id) REFERENCES regions(region_id));
```

ÄNDERN DER TABELLENSTRUKTUR

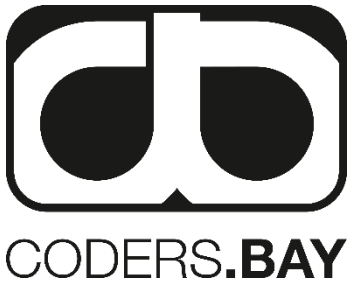


Nachträgliche Änderungen mit `ALTER TABLE`

BSP:

```
ALTER TABLE locations ADD (inhabitants INT(5));
ALTER TABLE locations MODIFY inhabitants INT(12);
ALTER TABLE locations DROP COLUMN inhabitants;
ALTER TABLE employees ADD FOREIGN KEY(manager_id) REFERENCES
employees(employee_id));
```

ÄNDERN DER TABELLENSTRUKTUR



Nachträgliche Löschung mit `DROP TABLE`

BSP:

```
DROP TABLE locations;
DROP TABLE locations CASCADE;
```

Mit CASCADE werden ‘abhängige’ Objekte ebenfalls gelöscht

DATA MANIPULATION LANGUAGE

HINZUFÜGEN VON DATEN

CODERS.BAY

```
INSERT
INTO countries (country_id, country_name, region_id)
VALUES ('AT', 'Austria', 1);
```

VERÄNDERN VON DATEN

CODERS.BAY

```
UPDATE countries
SET region_id = 2
WHERE country_name = ‘Austria’;
```

LÖSCHEN VON DATEN

CODERS.BAY

```
DELETE
FROM countries
WHERE country_name = ‘Austria’;
```

ENDE

CODERS.BAY