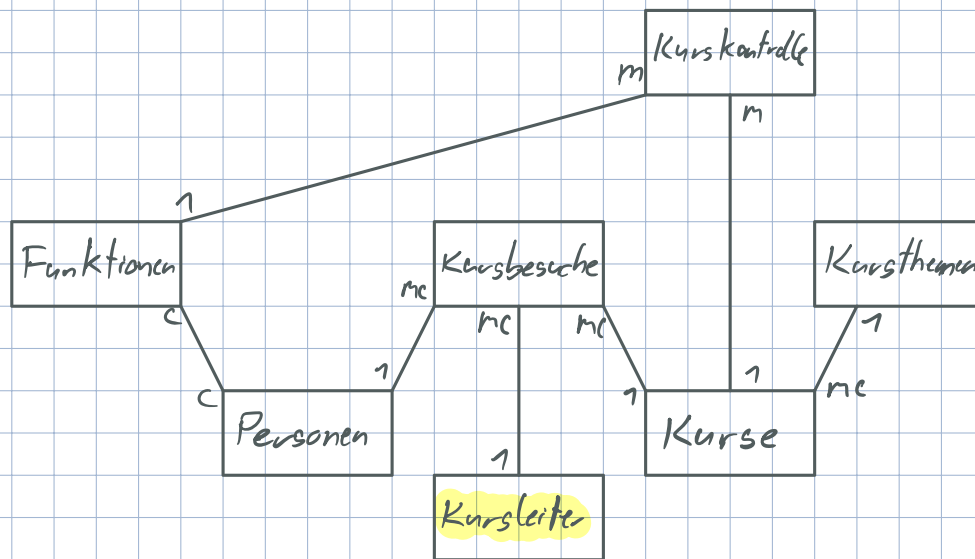


SELECT [DISTINCT] { * | Attributliste | mathematische Ausdrücke }

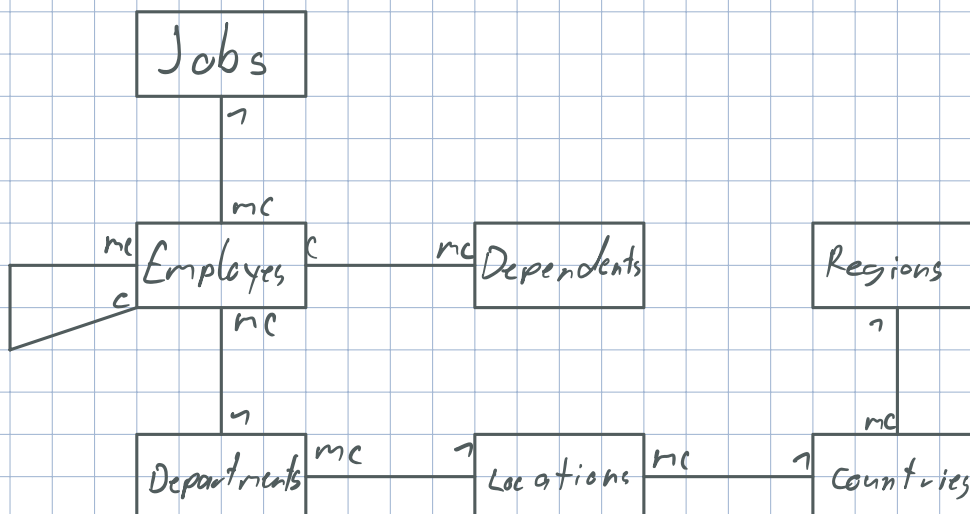
Bezeichner

FROM Tabelle 1 Bezeichner 1, Tabelle 2 Bezeichner 2, ...
[WHERE Bedingungen]
[GROUP BY Attributliste] [HAVING Bedingungen]
[ORDER BY Attributliste] [ASC | DESC]

Bsp 1:



Bsp 2:



Einfache Abfragen

SELECT *

← Attribute auswählen

FROM Kursthemen ;

↳ Ergebnis

TMr	Themengebiet
1	Sicherheit und Umweltschutz
2	...
3	...
4	...
5	...
6	...

SELECT PNr, Name, Vorname

← Attribute

(Lohnstufe - 1) * 10 000 + 60 000

Salary

Alias: Das den Ergebnisspalt

FROM Personen;

Angabe:

PNr	Name	Vorname	Salary
1001	Steffen	Felix	100000
...
...
...

zählt Zeilen des Ergebnis



SELECT COUNT(*), MIN(Datum) Erster,

MAX(Datum) letzter

FROM Kursbesuche;

Gruppenfunktion: wird auf alle Datensätze d. Ergebnisses angewendet. COUNT, MIN, MAX, SUM, AVG, ..

Ausgabe:

Anzahl	Erster	Letzter
14	07-AUG-08	25-AUG-08

SELECT COUNT (DISTINCT KNr) Verschiedene_Kurse
FROM KURSBESUCHE;

Verschiedene Kurse
7

Bedingungen

SELECT PNr, Name, Vorname

① FROM Personen

② WHERE FNr = 1;

↑
Abarbeitungs-
Reihenfolge

Bedingung

Ausgabe:

PNr	Name	Vorname
132442	Osswald	Kurt
...
...

SELECT PNr, Name, Vorname

FROM Personen

WHERE Lohnstufe >= 5

AND (For = 2 OR FNr = 3)

AND NOT (Name = 'Steffen');

Datensätze sortieren

SELECT *

FROM Funktionen

ORDER BY Funktionen; ASC/DESC

FNr	Funktionen
4	Bereichsleiter
...	...
...	...
...	...

SELECT PNr, KNr, Datum

FROM Kursbesuch

ORDER BY PNr, ASC, KNr, ASC, Datum, DESC

PNr	KNr	Datum
1001	245	23-JUN-08
1001	255	27-JUL-08
1001	412	07-AUG-08
1001	454	12-JAN-07

Datensätze Gruppieren

Personen			FNr	Salary	
PNr	VN	NN			
1	~	~	3	~	→
2	~	~	3	~	
3	~	~	4	~	
4	~	~	5	~	
6	~	~	4	~	
7	~	~	4	~	
8	~	~	3	~	
					<u>FNr</u>
					3
					4
					5

Braucht Aggregationsfunktionen(SUM, COUNT, usw.), da man aus allen Werten einen Wert bilden muss.

SELECT **FNr**, **COUNT**(FNr) Anzahl, **AVG**(Lohnstufe-1)*10000+(2000)
 DSalary

FROM Personen

GROUP BY **FNr**

ORDER BY

FNr	Anzahl	DSalary
5	1	100000
4	2	125000
3	3	93333,3
2	2	95000
1	3	63333,3

In Verbindung mit dem Schlüsselwort „GROUP BY“ gibt es noch das Schlüsselwort „HAVING“, welches das Definieren von Gruppenbedingungen ermöglicht. Im Gegensatz zu „WHERE“ werden die mit „HAVING“ angegebenen Bedingungen nicht auf einzelne Datensätze, sondern auf Datensatzgruppen angewendet.

SELECT PNr, COUNT(KNr) Anzahl

FROM Kursbesuche

WHERE Datum >= '01-SEPT-07'

GROUP BY PNr

HAVING COUNT(KNr) > 1;

Kundenbesuche

...	PNr	KNr	...
	1	9	
	1	10	
	1	11	
	2	9	
	2	10	
	3	9	



Verschachtelte Abfragen

SELECT KNr, Kursbezeichnung
 FROM Kurse

WHERE KNr IN (SELECT KNr
 FROM KURSBEsuche

WHERE PNr = (SELECT
 FROM
 WHERE
 AND

PNr
 Personen
 Name = 'Steffen'
 Vorname = 'Felix')

ORDER BY Kursbezeichnung

KNr	Kursbezeichnung
255	Datenbanken
454	Elektrotechnische Aufladung
245	Kostenrechnung
412	Tabellealkulation

```

SELECT FNr, Name, Vorname, Lohnstufe
FROM Personen
WHERE (FNr, Lohnstufe) IN
      (SELECT FNr, MAX(Lohnstufe)
       FROM Personen
       GROUP BY FNr)
ORDER BY FNr DESC, Name ASC;

```





FNr	Name	Vorname	Lohnstufe
5	Steiner	Rene	5
4	Huber	Walter	8
3	Steffen	Felix	5
2	Meier	Hans	5
1	Osswald	Kurt	2

JOINS

Employee					
Id	Name	jid		Id	Beschr
1	Max	1	x	1	Maurer
2	Toni	2		2	Dachdecker
3	Hans	3			

Id	Name	jid	Id	Beschr
1	Max	1	1	Maurer
1	Max	1	2	Dachdecker
2	Toni	1	1	Maurer
2	Toni	1	2	Dachdecker
3	Hans	2	1	Maurer
3	Hans	2	2	Dachdecker

Es gibt folgende Arten von Joins:

- INNER 
- LEFT 
- RIGHT 
- CROSS 

Tabellen verknüpfen (Joining)

A	B	C		D	E	F
I	~	~	X	α	~	~
II	~	~		β	~	~

Cross

A	B	C	D	E	F
I	~	~	α	~	~
I	~	~	β	~	~
I	~	~	α	~	~
II	~	~	β	~	~

SELECT * FROM Funktionen, Personen

A	B	C'		C''	E	F
I	~		X	α	~	~
II	~			β	~	~
III	~					

WHERE C' = C''

Inner

A	B	C'	C''	E	F
I	~	α	α	~	~
II	~	α	α	~	~
III	~	β	β	~	~

Bsp.

SELECT P.Nr, Name, Vorname, Funktion
FROM Personen, Funktionen
WHERE Personen.FNr = Funktionen.FNr;

Ausgabe:

PNr	Name	Vorname	Funktion
132442	Osswald	Kurt	Vorarbeiter
232452	Müller	Hugo	Vorarbeiter
345678	Metzger	Paul	Vorarbeiter
334643	Meier	Hans	Meister
344556	Scherrer	Daniel	Meister
100001	Steffen	Felix	Chemiker
567231	Schmid	Beat	Chemiker
625342	Gerber	Roland	Chemiker
233456	Müller	Franz	Bereichsleiter
845622	Huber	Walter	Bereichsleiter
345727	Steiner	René	Informatiker

```

SELECT PNr, Name, Vorname, COUNT(KNr) Anzahl(
FROM Kurskontrolle A, Personen B
WHERE (PNr | KNr) NOT IN ← join
      (SELECT PNr, KNr
       FROM Kursbesuche)
AND A.FNr = B.FNr join Bedingung
GROUP BY PNr, Name, Vorname
HAVING COUNT(KNr) > 3
ORDER BY PNr;

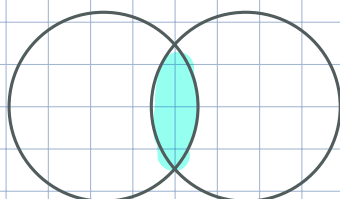
```

Die 4 Join Arten

A	B	C	D	E	F
I	~	1	a	1	~
II	~	2	b	b	~
III	~	2	c	c	~
IV	~	99	d	d	~

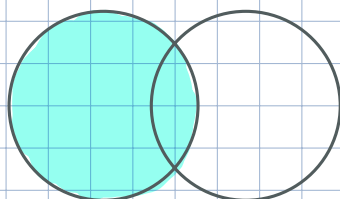
X
C=E

inner-join



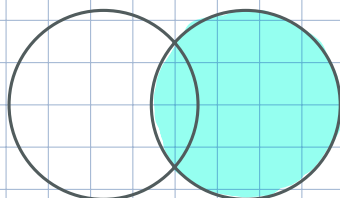
A	B	C	D	E	F
I	~	1	a	1	~
II	~	2	b	2	~
III	~	2	b	2	~

left-join



A	B	C	D	E	F
I	~	1	a	1	~
II	~	2	b	2	~
III	~	2	b	2	~
IV	~	99	NULL	NULL	NULL

right-join



A	B	C	D	E	F
I	~	1	a	1	~
II	~	2	b	2	~
III	~	2	b	2	~
NULL	NULL	NULL	d	55	~

cross-join

→ jedes Tupel mit jedem

SQLite Query für Inner-Join:

```
SELECT  
INNER JOIN jobs  
ON employees.job_id = jobs.job_id  
FROM employees
```