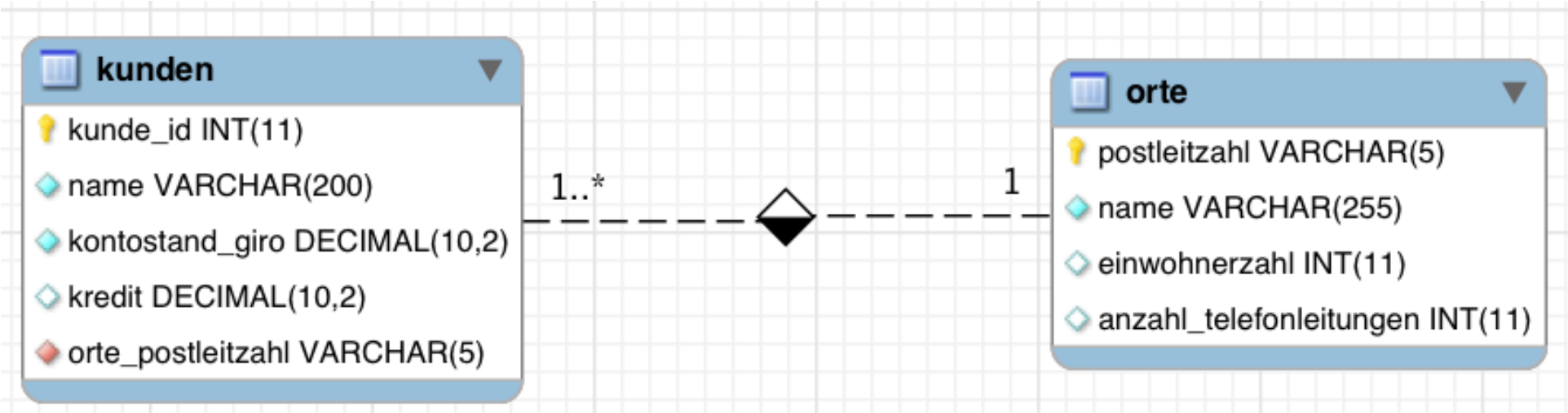
MySQL: Einfache Aggregatfunktionen

(= Funktionen, die etwas zusammenfassen)

Beispieldatenbank "Kunden"

kunden (kunde_id, name, ⇧orte_postleitzahl,
kontostand_giro, kredit)
orte (postleitzahl, name, einwohnerzahl,
anzahl_telefonleitungen)



Beispieldatenbank "Kunden"

kunden (kunde_id, name, ⇧ort_postleitzahl, kontostand_giro, kredit)
orte (postleitzahl, name, einwohnerzahl, anzahl_telefonleitungen)

postleitzahl	name	einwohnerzahl	anzahl_telefonleitungen
80995	München	1000000	385
79312	Emmendingen	40000	12
79111	Freiburg	280000	195
20095	Hamburg	2000000	1004

kunde_id	name	ort_postleitzahl	kontostand_giro	kredit
1	John	79111	182.00	-430320.22
2	Herbert	79312	10291.32	-10000.00
3	Sabina	79312	-253.21	-3205.32
4	Mary	79111	-832.01	NULL
5	Heinrich	79111	15302.85	0.00
6	Usal	80995	23012.21	NULL
7	Johannes	80995	159.31	0.00
8	Carla	79312	503.06	-15302.68
9	Ludowika	79111	25201.07	-82213.99
10	Niemand	99999	-5021.30	-3024.21

(Code)

für Copy-Paste

```
DROP DATABASE IF EXISTS `kunden`;
CREATE DATABASE IF NOT EXISTS `kunden` /*!40100 DEFAULT CHARACTER SET latin1 */;
USE `kunden`;
-- MySQL dump 10.13  Distrib 5.7.12, for osx10.9 (x86_64)
--
-- Host: 127.0.0.1    Database: Kunden
-- ------------------------------------------------------
-- Server version	5.5.38

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `kunden`
--

DROP TABLE IF EXISTS `kunden`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `kunden` (
  `kunde_id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  `ort_postleitzahl` varchar(5) NOT NULL,
  `bonitaet_pkt` decimal(10,2) NOT NULL,
  `kredit` decimal(10,2) DEFAULT NULL,
  PRIMARY KEY (`kunde_id`),
  KEY `FK_kunde_ort` (`ort_postleitzahl`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `kunden`
--

LOCK TABLES `kunden` WRITE;
/*!40000 ALTER TABLE `kunden` DISABLE KEYS */;
INSERT INTO `kunden` VALUES (1,'John','79111',182.00,-400320.22),(2,'Herbert','79312',10291.32,-10000.00),(3,'Sabine','79312',-253.21,-205.32),(4,'Mary','79111',-822.33,NULL),(5,'Heinrich','79111',15302.85,0.00),(6,'Karl','80995',25012.21,NULL),(7,'Johannes','80995',159.31,0.00),(8,'Carla','79312',503.08,-15302.68),(9,'Ludowika','79111',25201.07,-8213.99),(10,'Diamond','99999',-9221.30,-3524.21);
/*!40000 ALTER TABLE `kunden` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `orte`
--

DROP TABLE IF EXISTS `orte`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `orte` (
  `postleitzahl` varchar(5) NOT NULL,
  `name` varchar(255) NOT NULL,
  `einwohnerzahl` int(11) DEFAULT NULL,
  `anzahl_telefonleitungen` int(11) DEFAULT NULL,
  PRIMARY KEY (`postleitzahl`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `orte`
--

LOCK TABLES `orte` WRITE;
/*!40000 ALTER TABLE `orte` DISABLE KEYS */;
INSERT INTO `orte` VALUES ('20095','Hamburg',2000000,1004),('79111','Freiburg',280000,190),('79312','Emmendingen',40000,12),('80995','München',1000000,385);
/*!40000 ALTER TABLE `orte` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2016-11-02 20:20:40
```

Aggregatfunktionen

Ähnlich wie in Excel:

- COUNT() → Datensätze zählen
- SUM() → Summe *schon bekannt*
- AVG() → Durchschnitt *schon bekannt*
- MAX() → Maximum
- MIN() → Minimum

gute Erklärung im Web:
http://www.teialehrbuch.de/Kostenlose-Kurse/SQL/14750-Aggregatfunktionen.html

MIN(), MAX()

```
SELECT
    *
FROM
    orte
```

MIN – gibt den kleinsten Wert einer Spalte zurück
MAX – gibt den höchsten Wert einer Spalte zurück

postleitzahl	name	einwohnerzahl	anzahl_telefonleitung...
80995	München	1000000	385
79312	Emmendingen	40000	12
79111	Freiburg	280000	195
20095	Hamburg	2000000	1004

MIN(), MAX()

```
SELECT
   MIN(anzahl_telefonleitungen)
FROM
   orte
```

postleitzahl	name	einwohnerzahl	anzahl_telefonleitung...
80995	München	1000000	385
79312	Emmendingen	40000	12
79111	Freiburg	280000	195
20095	Hamburg	2000000	1004

MIN(), MAX()

```
SELECT
   MIN(anzahl_telefonleitungen) AS min,
   MAX(anzahl_telefonleitungen) max (*)

FROM
   orte
```

postleitzahl	name	einwohnerzahl	anzahl_telefonleitung...
80995	München	1000000	385
79312	Emmendingen	40000	12
79111	Freiburg	280000	195
20095	Hamburg	2000000	1004

min	max
12	1004

(*) Erinnerung:
AS kann
weggelassen werden

MIN(), MAX()

Achtung: Fehler!

```
SELECT
   MIN(anzahl_telefonleitungen), name
FROM
   orte
```

postleitzahl	name	einwohnerzahl	anzahl_telefonleitung...
80995	München	1000000	
79312	Emmendingen	40000	12
79111	Freiburg	280000	195
20095	Hamburg	2000000	1004

min(anzahl_telefonleitungen)	name
12	München

MIN/MAX fasst alle Datensätze zu EINEM Ergebnis zusammen (Aggregat).
Zusätzliche Auswahl eines Attributs zeigt hier den ersten Wert an ("München")

MIN(), MAX()

```
SELECT *, MIN(kontostand_giro)
FROM kunde
```

Achtung: Fehler!

Das geht nicht!

kunde_id	name	ort_postleitzahl	kontostand_giro	kredit	MIN(kontostand...
1	John	79111	182.00	-430320.22	-5021.30

Da Aggregierungs-Funktionen in WHERE-Clauses nicht erlaubt sind, ist eine Subquery nötig oder Kreativität:

```
SELECT *
FROM kunde
ORDER BY kontostand_giro ASC
LIMIT 1
```

COUNT(): Anzahl Datensätze

```
SELECT * FROM orte
```

postleitzahl	name	einwohnerzahl	anzahl
80995	München	1000000	385
79312	Emmendingen	40000	12
79111	Freiburg	280000	195
20095	Hamburg	2000000	1004

Wie viele Zeilen sind das?
(oder: Wie viele Datensätze bekommen wir als Ergebnis?)

COUNT(): Anzahl Datensätze

```
SELECT * FROM orte
```

postleitzahl	name	einwohnerzahl	anzahl
80995	München	1000000	385
79312	Emmendingen	40000	12
79111	Freiburg	280000	195
20095	Hamburg	2000000	1004

Wie viele Zeilen sind das?
(oder: Wie viele Datensätze bekommen wir als Ergebnis?)

```
SELECT
    COUNT(*) AS 'Anzahl aller Orte'
FROM
    orte
```

Anzahl aller Orte
4

COUNT(): Anzahl Datensätze

```
SELECT * FROM orte
```

postleitzahl	name	einwohnerzahl	anzahl
80995	München	1000000	385
79312	Emmendingen	40000	12
79111	Freiburg	280000	195
20095	Hamburg	2000000	1004

COUNT() gibt die Anzahl der Ergebnis-Datensätze zurück

`SELECT COUNT(*) FROM xy` gibt also die Anzahl der Ergebniszeilen von `SELECT * FROM xy` zurück.

```
SELECT
    COUNT(*) AS 'Anzahl aller Orte'
FROM
    orte
```

Anzahl aller Orte
4

COUNT(): Anzahl Datensätze

```
SELECT
     *
FROM
     kunden;
```

kunde_id	name	ort_postleitz...	kontostand_giro	kredit
1	John	79111	182.00	-430320.22
2	Herbert	79312	10291.32	-10000.00
3	Sabina	79312	-253.21	-3205.32
4	Mary	79111	-832.01	NULL
5	Heinrich	79111	15302.85	0.00
6	Usal	80995	23012.21	NULL
7	Johannes	80995	159.31	0.00
8	Carla	79312	503.06	-15302.68
9	Ludowika	79111	25201.07	-82213.99
10	Niemand	99999	-5021.30	-3024.21

```
SELECT
     COUNT(*) 'Anzahl aller Kunden'
FROM
     kunden;
```

Anzahl aller Kunden
10

COUNT(): Anzahl Datensätze

```
SELECT
     kredit
FROM
     kunden;
```

kredit
-430320.22
-10000.00
-3205.32
NULL
0.00
NULL
0.00
-15302.68
-82213.99
-3024.21

COUNT(): Anzahl Datensätze

```
SELECT
     kredit
FROM
     kunden;
```

kredit
-430320.22
-10000.00
-3205.32
NULL
0.00
NULL
0.00
-15302.68
-82213.99
-3024.21

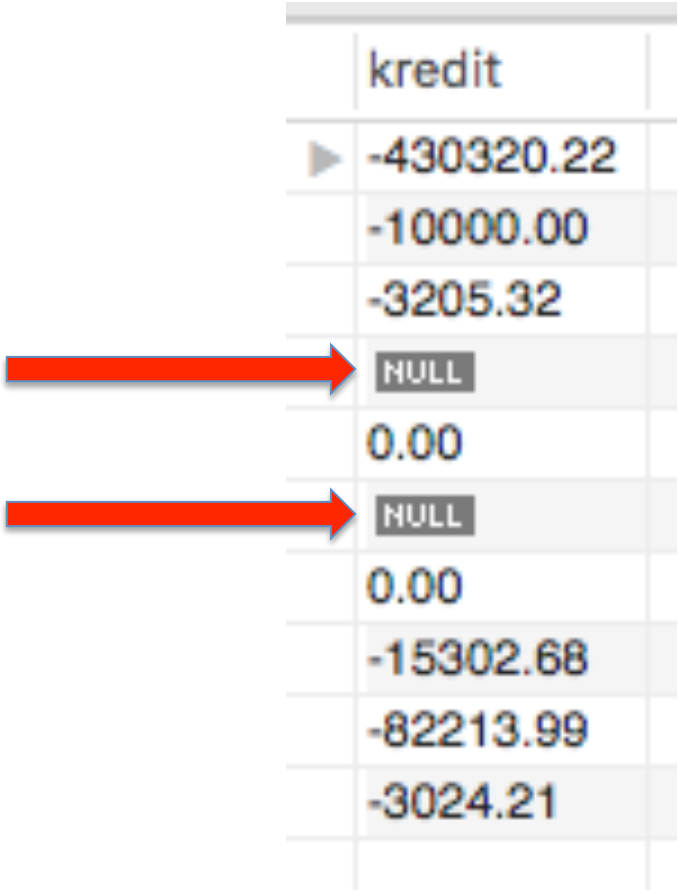
Was ergibt die Abfrage

```
SELECT
     COUNT(kredit)
FROM
     kunden;
```

?

COUNT(): Anzahl Datensätze

```
SELECT
     kredit
FROM
     kunden;
```



kredit
-430320.22
-10000.00
-3205.32
NULL
0.00
NULL
0.00
-15302.68
-82213.99
-3024.21

10 Ergebniszeilen
(2 davon enthalten keinen Wert – NULL)

COUNT(): Anzahl Datensätze

```
SELECT
     COUNT(kredit)
FROM
     kunden;
```

kredit
-430320.22
-10000.00
-3205.32
NULL
0.00
NULL
0.00
-15302.68
-82213.99
-3024.21

COUNT(kredit)
8

COUNT(): Anzahl Datensätze

```
SELECT
    COUNT(kredit)
FROM
    kunden;
```

ergibt 8
(alle Datensätze, die einen Wert haben).

kredit
-430320.22
-10000.00
-3205.32
NULL
0.00
NULL
0.00
-15302.68
-82213.99
-3024.21

Wir wollen aber die Kunden, deren Kredit 0.00 ist, nicht mitzählen.
Erweitern Sie die Abfrage entsprechend.

COUNT(): Anzahl Datensätze

```
SELECT
    COUNT(kredit)
FROM
    kunden
 WHERE
    kredit < 0; -- nur negative Werte

...
 WHERE
    kredit != 0; -- ungleich 0

...
 WHERE
    kredit < 0 OR kredit > 0;
      -- ungleich 0, wie oben
```

kredit
-430320.22
-10000.00
-3205.32
NULL
0.00
NULL
0.00
-15302.68
-82213.99
-3024.21

COUNT(kredit)
6