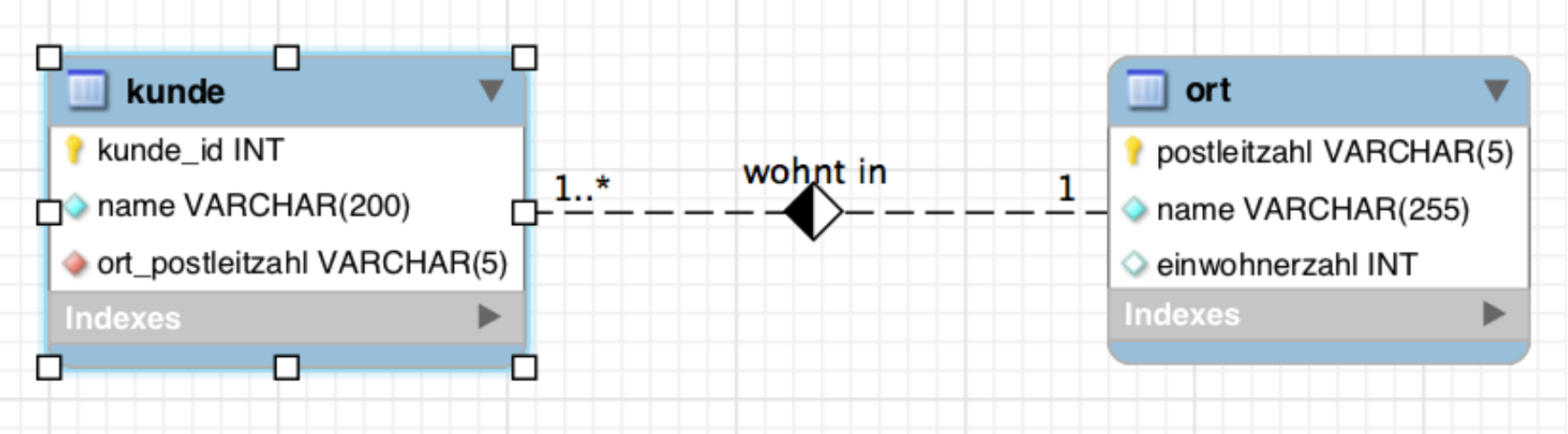
MySQL:
SELECT-Abfragen über mehrere Tabellen (JOINs)

Grau hinterlegte Folien enthalten Detailthemen und sind nicht superwichtig.

Beispiel

kunden (<u>kunde_id</u>, name, ⇧ort_postleitzahl)
orte (<u>postleitzahl</u>, name, einwohnerzahl)



Beispiel

kunden (kunde_id, name, ⇧ort_postleitzahl)
orte (postleitzahl, name, einwohnerzahl)

kunde_id	name	ort_postleitzahl
1	John	7911
2	Herbert	7931
3	Sabina	7931
4	Mary	7911
5	Heinrich	7911
6	Usal	8099
7	Johannes	80995
8	Carla	79312
9	Ludowika	79111
10	Niemand	99999

postleitzahl	name	einwohne
80995	München	1000000
79312	Emmendingen	40000
79111	Freiburg	280000
20095	Hamburg	2000000

Kunde "Niemand" = ungültiger Ort
Ort Hamburg = keine Kunden

(Code)

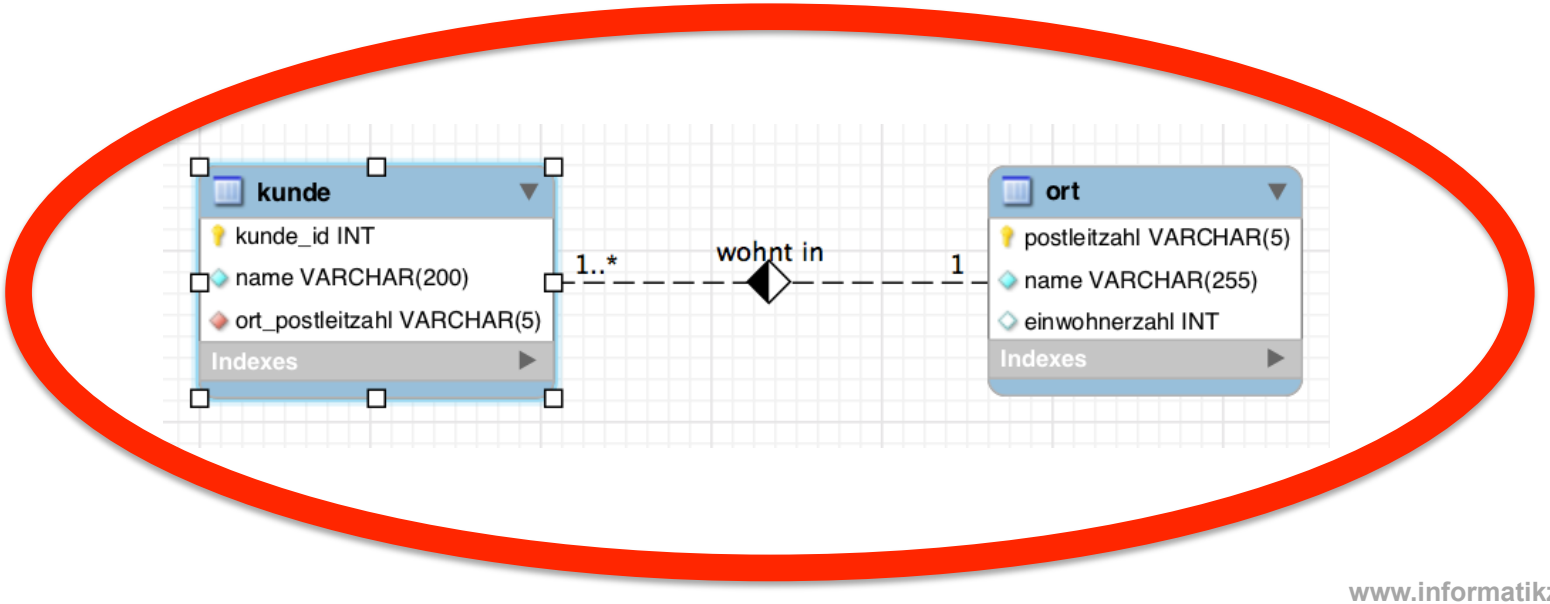
für Copy-Paste - _kunden-einfach-dump.sql

```
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 */;
/*!40101 SET NAMES utf8 */;
/*!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,
  PRIMARY KEY (`kunde_id`),
  KEY `fk_kunde_ort` (`ort_postleitzahl`)
) ENGINE=MyISAM 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'),(2,'Herbert','79012'),(3,'Sabine','79012'),(4,'Mary','79111'),(5,'Heinrich','79111'),(6,'Carl','80995'),(7,'Johannes','80995'),(8,'Carla','79012'),(9,'Ludmilla','79111'),(10,'Richard','99999');
/*!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,
  PRIMARY KEY (`postleitzahl`)
) ENGINE=MyISAM 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 ('80995','München',1000000),('79312','Emmendingen',40000),('79111','Freiburg',280000),('20095','Hamburg',2000000);
/*!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 20:24:13
```

Abfrage über mehrere Tabellen

```
SELECT * FROM kunde JOIN ort
```

→ eine Abfrage über kunde UND ort wird ausgeführt



Alternative Syntax:

EXPLIZITE SCHREIBWEISE:

```
SELECT * FROM kunde JOIN ort
SELECT * FROM ort JOIN kunde
```

IMPLIZITE SCHREIBWEISE:

```
SELECT * FROM ort, kunde
```

ALS INNER JOIN

```
SELECT * FROM ort INNER JOIN kunde
```

(Bewirkt exakt das Gleiche wie JOIN;
Existenzgrund: "syntactic sugar": http://en.wikipedia.org/wiki/Syntax_sugar)

Implizite Schreibweise

EXPLIZITE SCHREIBWEISE:

```
SELECT * FROM kunde JOIN ort
```

IMPLIZITE SCHREIBWEISE:

```
SELECT * FROM ort, kunde
```

Im Folgenden benutzen wir immer die implizite Schreibweise!

Ergebnis: Kartesisches Produkt (Kreuzprodukt)

```
SELECT * FROM kunde, ort
```

Jeder Datensatz der einen Tabelle wird mit
jedem Datensatz der anderen Tabelle kombiniert! (= sinnloses Ergebnis)

kunde_id	name	ort_postleitzahl	postleitzahl	name	einwohnerza
1	John	79111	80995	München	1000000
1	John	79111	79312	Emmendingen	40000
1	John	79111	79111	Freiburg	280000
1	John	79111	20095	Hamburg	2000000
2	Herbert	79312	80995	München	1000000
2	Herbert	79312	79312	Emmendingen	40000
2	Herbert	79312	79111	Freiburg	280000
2	Herbert	79312	20095	Hamburg	2000000
3	Sabina	79312	80995	München	1000000

Beziehungen zwischen den Tabellen werden nicht beachtet!

Einschränkung auf sinnvolle Datensätze: Equi-Join

```
SELECT * FROM kunde, ort
  WHERE ort_postleitzahl = postleitzahl
```

Ergebnis wird eingeschränkt auf die Gleichheit im Attribut *postleitzahl*

kunde_id	name	ort_postleitzahl	postleitzahl	name	einwohnerzahl
6	Usal	80995	80995	München	1000000
7	Johannes	80995	80995	München	1000000
2	Herbert	79312	79312	Emmendingen	40000
3	Sabina	79312	79312	Emmendingen	40000
8	Carla	79312	79312	Emmendingen	40000
1	John	79111	79111	Freiburg	280000
4	Mary	79111	79111	Freiburg	280000
5	Heinrich	79111	79111	Freiburg	280000
9	Ludowika	79111	79111	Freiburg	280000

Alle Datensätze einer Tabelle auf jeden Fall ausgeben: LEFT JOIN

```
SELECT * FROM kunde LEFT JOIN ort
   ON kunde.ort_postleitzahl = ort.postleitzahl
```

Alle Datensätze der LINKEN Tabelle ("kunde") werden ausgegeben, auch wenn keine Entsprechung in der rechten Tabelle. In diesem Fall wird als Ergebnis der rechten Tabelle NULL ausgegeben.

kunde_id	name	ort_postleitzahl	ort_postleitzahl	name	einwohnerza
1	John	79111	79111	Freiburg	280000
2	Herbert	79312	79312	Emmendingen	40000
3	Sabina	79312	79312	Emmendingen	40000
4	Mary	79111	79111	Freiburg	280000
5	Heinrich	79111	79111	Freiburg	280000
6	Usal	80995	80995	München	1000000
7	Johannes	80995	80995	München	1000000
8	Carla	79312	79312	Emmendingen	40000
9	Ludowika	79111	79111	Freiburg	280000
10	Niemand	99999	NULL	NULL	NULL

Alle Datensätze einer Tabelle auf jeden Fall ausgeben: LEFT JOIN

```
SELECT * FROM kunde LEFT JOIN ort
   ON kunde.ort_postleitzahl = ort.postleitzahl
```

Alle Datensätze der LINKEN Tabelle ("kunde") werden
ausg Tabelle.
In die usgegeben.

ACHTUNG:
BEI LEFT JOIN
statt WHERE immer
ON verwenden!

kunde_id					einwohnerza
1				g	280000
2				ndingen	40000
3				ndingen	40000
4				g	280000
5				g	280000
6				en	1000000
7				en	1000000
8	Carla	79312	79312	Emmendingen	40000
9	Ludowika	79111	79111	Freiburg	280000
10	Niemand	99999	NULL	NULL	NULL

Alle Datensätze einer Tabelle auf jeden Fall ausgeben: LEFT JOIN

```
SELECT * FROM ort LEFT JOIN kunde
   ON ort.postleitzahl = kunde.ort_postleitzahl
```

(**nun steht "ort" links**, damit werden alle Orte ausgegeben, auch die, in denen keiner wohnt)

ort_postleitzahl	name	einwohnerzahl	kunde_id	name	ort_postleitza
80995	München	1000000	6	Usal	80995
80995	München	1000000	7	Johannes	80995
79312	Emmendingen	40000	2	Herbert	79312
79312	Emmendingen	40000	3	Sabina	79312
79312	Emmendingen	40000	8	Carla	79312
79111	Freiburg	280000	1	John	79111
79111	Freiburg	280000	4	Mary	79111
79111	Freiburg	280000	5	Heinrich	79111
79111	Freiburg	280000	9	Ludowika	79111
20095	Hamburg	2000000	NULL	NULL	NULL

RIGHT JOIN

funktioniert wie LEFT JOIN (nur andersrum)

☺

Abfrage von mehr als 2 Tabellen

```
SELECT * FROM kunde k, ort o, bundesland b

WHERE
  k.postleitzahl = o.postleitzahl
  AND
  o.bundesland   = b.bundesland
```

Tipp:
Bei n Tabellen haben wir immer n-1
Primärschlüssel-Fremdschlüssel-Einschränkungen

Im Beispiel oben: 3 Tabellen, 2 SELECT-Bedingungen