Um den **Beziehungstyp zu bestimmen**, geht man wie folgt vor:

1) **Erstens** Als Ausgangslage braucht es zwei Entitätsmengen und eine Beschreibung der Beziehung (Verb, Substantiv oder kurzer Satz). Beispiel: Entitätsmenge 1 = Kunde, Entitätsmenge 2 = Ort, Beziehung = wohnt bzw. wohnen
2) **Zweitens** Um die Kardinalität auf der rechten Seite (Ort) zu bestimmen, geht man von der Menge aller Kunden aus: Jeder Kunde wohnt an genau einem Ort **->** Kardinalität = 1.
3) **Drittens** Um die Kardinalität auf der linken Seite (Kunde) zu bestimmen, geht man von der Menge aller Orte aus: An jedem Ort wohnen 0, 1 oder mehrere Kunden **->** Kardinalität = mc.
4) **Demnach** ist die Beziehung mc:1

ANOMALIEN sind: **Update-Anomalie**, wenn gleiche Daten unterschiedlich benannt werden und dadurch bei Updates nur Teile der DB verändert werden. **Delete-Anomalie**, wenn beim Löschen einer Entität ungewollt auch andere Informationen verloren gehen. **Insert-Anomalie**, wenn Attribute fehlen, mehrere Primärschlüssel dieselbe Information tragen oder ein Objekt mehrere Schlüssel trägt.

ABHÄNGIGKEITEN

RNr	PNr	Datum	Name	KNr	Adresse	Artikel	ANr	Anz	Prei
123	1	29.01.2020	Marc Muster	11	Talweg 11, 3000 Bern	Monitor	2-023	10	200
123	2	29.01.2020	Marc Muster	11	Talweg 11, 3000 Bern	Maus	4-023	12	25 C
123	3	29.01.2020	Marc Muster	11	Talweg 11, 3000 Bern	Bürostuhl	5-023	1	450
124	1	30.01.2020	Erika Muster	12	Bergweg 21, 3000 Bern	Notebook	1-023	2	1200
124	2	30.01.2020	Erika Muster	12	Bergweg 21, 3000 Bern	Maus	4-023	2	25 C

1) **funktionale Abhängigkeit** liegt dann vor, wenn ein Nichtschlüsselattribut nur von einem Teil der Attribute eines zusammengesetzten Schlüsselkandidaten funktional abhängig ist. Beispiel: Das Datum ist funktional abhängig von der Rechnungsnummer *RNr*, nicht aber vom gesamten Schlüssel *RNr* und *PNr*.
2) **vollständig funktionale (volle) Abhängigkeit** liegt dann vor, wenn ein Nichtschlüsselattribut nicht nur von einem Teil der Attribute eines zusammengesetzten Schlüsselkandidaten funktional abhängig ist, sondern von allen Teilen des Primärschlüssels (bzw von mehreren Nichtschlüsselattributen) . Die vollständig funktionale Abhängigkeit wird mit der zweiten Normalform erreicht. Beispiel: Der Artikel ist vollständig funktional abhängig vom gesamten Primärschlüssel *RNr* und *PNr*.
3) **transitive Abhängigkeit** liegt dann vor, wenn ein Nichtschlüsselattribut funktional abhängig ist von einem andern Nichtschlüsselattribut. Es ist transitiv oder indirekt abhängig vom Primärschlüssel. Beispiel: Name und Adresse sind abhängig von der Kundennummer *KNr*. Somit sind diese Attribute transitiv abhängig vom Schlüsselkandidaten.

NORMALFORMEN- Definition Nullte Normalform: Eine Tabelle befindet sich in der nullten Normalform, wenn alle Datenelemente der realen Welt darin zusammengefasst und aufgelistet sind. **erste Normalform:** wenn alle Daten atomar vorliegen und alle Tabellenspalten gleichartige Werte enthalten, also muss jedes Attribut atomar sein. **Definition zweite Normalform:** wenn sie sich in der ersten Normalform befindet und zudem jedes einzelne Nichtschlüsselattribut vom Primärschlüssel voll funktional abhängig ist. Also darf kein Attribut voll abhängig sein! **Definition dritte Normalform:** wenn er sich in der zweiten Normalform befindet und es zudem *keine transitiven Abhängigkeiten* gibt.

- **Nullte Normalform**: Eine Tabelle ist in der nullten Normalform, wenn alle Daten der realen Welt darin unstrukturiert aufgefasst sind. In so einer Tabelle können in einem Attributwert auch mehrere Informationen gelistet sein (was in einer DB nicht vorkommen sollte (atomarität)).
- **Erste Normalform**: Wenn alle Daten atomar vorliegen (pro Attributwert nur ein Wert), ist die Tabelle in der ersten Normalform. Man trennt also alle Informationen in die kleinst mögliche Einheit auf (die Sinn ergibt) und gibt ihr eine eigene Spalte (macht sie zu einem eigenen Attribut)
- **Zweite Normalform**: Wenn die Attribute einer Tabelle nur von einem Primärschlüssel abhängig sind, ist sie automatisch bereits in der zweiten Normalform. Der Unterschied zur ersten Normalform besteht darin, dass jedes Attribut von den Schlüsseln abhängig ist. Es können jedoch immernoch Abängigkeiten von Attributen untereinander bestehen wie zb. die Abhängigkeit von Name zu Kundennummer. Um eine Tabelle in die zweite normalform umzuwandeln, muss man alle Nichtschlüsselattribute, die nicht vollständig vom Primärschlüssel abhängig sind, in seperate Tabellen umlagern
- **Dritte Normalform**: In Tabellen der dritten Normalform sind alle Attribute der Tabelle nur noch von ihrem Primärschlüssel und nicht mehr von anderen Attributen abhängig. Es werden sperate Tabellen gemacht, die nur dafür da sind, die Schlüssel verschiedener Tabellen (Einkäufe, Kunden) zu verbinden.

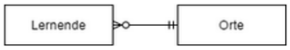
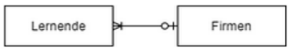
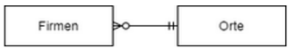
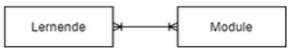
SQL-BEFEHLE: LIMIT<Zahl>-Wie viele Werte zurückgegeben werden einschränken. DISCTINCT-zeige jeden Wert nur einmal an. <attribut>||«»||<attribut>-gibt die beiden Attribute innerhalb einer Spalte aus.

```
SELECT Strasse, PLZ, Ort FROM Lernende l
  JOIN Orte o ON l.ort_id = o.id
UNION SELECT Strasse, PLZ, Ort FROM Firmen f
  JOIN Orte o ON f.ort_id = o.id
  ORDER BY Ort, Strasse;
```

SUM(),AVG(),MIN(),MAX(), ORDER BY <attribut>, ASC/DESC. WHERE-Befehl zum Filtern. BETWEEN <wert1> AND <wert2>. LIKE-lässt einen Platzhalter wie «_» (ein Zeichen) oder «%» (mehrere Zeichen – vorname like «b%»-findet vornamen, die mit b beginnen) verwenden. GROUP BY-Bei Aggregatsfunktionen, die normalerweise einen einzelnen Wert zurückgeben, kann man mit GROUP BY diese Werte pro Attribut xyz ausgeben lassen (zb sum(id) pro Ort).

Konzeptionelles Modell-Attribute, Entitätsmengen, Beziehungen. Datenobjekte identifizieren->Merkmale (Attribute) der Datenobjekte definieren->Von Datenobjekten zu Tabellen->Beziehungen zwischen Tabellen festlegen

Logisches Modell->Schlüssel, Datentypen, Zwischentabellen.

Jeder Lernende wohnt an 1 Ort. An jedem Ort wohnen 0, 1 oder mehrere Lernende	
Jeder Lernende ist in 0 oder 1 Firma angestellt. Jede Firma beschäftigt 1 oder mehrere Lernende	
Jede Firma befindet sich an 1 Ort. An jedem Ort befinden sich 0, 1 oder mehrere Firmen.	
Jeder Lernende besucht mehrere Module. Jedes Modul wird von mehreren Lernenden besucht.	

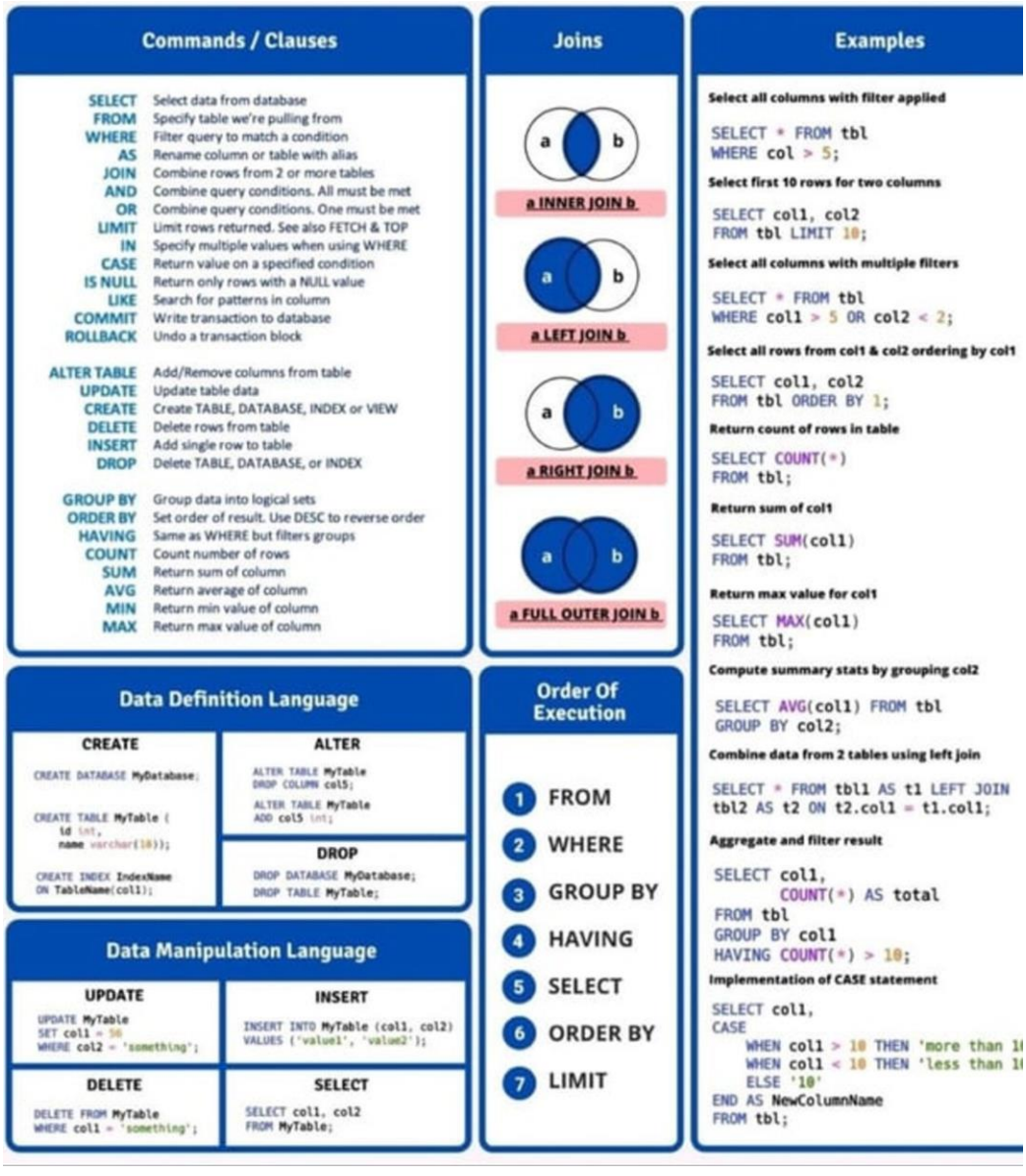
JOIN-IMMER ON VERWENDEN- Inner Join/Join: Gibt nur das zurück, was in beiden Datensätzen vorkommt. Das passiert nur, wenn nicht alle Datensätze die ON/WHERE Klausel erfüllen. Sonst wird alles retourniert. LEFT JOIN: Gibt alle Inhalte der linken Tabelle. Muss nur verwendet werden, wenn Inhalte der linken und der rechten Tabelle nicht übereinstimmen (fehlende fremdschlüssel können sich so trotzdem anzeigen lassen). UNION – Kombiniert zwei Tabellen, deren Attribute identisch sind. Wird wie ein weiteres Select nach dem ersten Select verwendet, mit Angabe der Attribute und allem.

```
SELECT l.Vorname, l.Nachname, l.Strasse, f.Firma
FROM Lernende AS l
LEFT JOIN Firmen AS f
ON l.firma_id = f.id
```

Füge Firmennamen zu den Lernenden hinzu, aber so, dass auch lernende ohne Firma angezeigt werden (left join)

```
SELECT count(*), jahr from lernende group by jahr
order by count(*) desc
```

Zähle die Anzahl Personen pro Jahr und sortiere nach Jahren absteigend



>,<,<= und => können auch bei Buchstaben verwendet werden.