Datenbanken

DBMS (= **D**aten**b**ank-**M**anagement **S**ystem) auch Datenbank Manager ist ein Informationssystem, das große Mengen von Daten auf permanente Medien beschreibt, speichert, manipuliert und wiedergewinnt.

Bei einem DBMS sind folgende Punkte zu beachten:

* Konsistenz (Daten dauerhaft verfügbar)
* Zugriff (Sicherheit und Zugriff nur durch berechtigte Benutzer)
* Effizienz der Datenhaltung (Redundanz vermindern)

Man spricht von einer Datenbank, wenn man von einem DBMS verwaltete Daten meint.[[1]](#footnote-1) Die Datenbank muss nicht elektronisch sein. Eine Datenbank beinhaltete Tabellen, die wiederum mit Daten befüllt sind.[[2]](#footnote-2)

Eine Datenbank kann, wenn man auf einer Website dynamisch eingegebene Daten, die sich in Tabellenform speichern lassen, speichern will, angewendet werden. Der Vorteil von einer Datenbank ist das mehrere Benutzer gleichzeitig darauf zugreifen können. Datenbanken werden von verschiedenen Update-Mechanismen unterstützt. Eine Schnittstelle wird benötigt, wenn der Webserver keinen Datenbank-Client enthält.[[3]](#footnote-3) Jas-t benutzt eine MySQL Datenbank, wo man mithilfe von einer Rest-Schnittstelle auf die Daten zugreift.

Probleme das Auftreten können

Eine RDBMS (=rationale DBMS) ist eine elektronische Datenbank, das transaktionale Tabellentypen unterstützt.[[4]](#footnote-4)

Integritätsprobleme können auftreten, wenn mehrere Benutzer gleichzeitig auf dieselben Daten zugreifen und diese manipulieren. Dass dies nicht aufritt muss sichergestellt werden das die gleichen Daten nicht zur selben Zeit verändert werden. Integritätsproblemen können mithilfe von Transaktionen vermieden werden.[[5]](#footnote-5) Eine Transaktion ermöglicht eine Menge von Operatoren die mit COMMIT gespeichert werden. Durch ROLLBACK können diese Operatoren rückgängig gemacht werden.[[6]](#footnote-6) Bei Datenschäden können Transaktionssystem neu erstellt werden und bei Systemabsturz können Daten von der letzten Sicherung wiederhergestellt werden, indem man das Transaktionsprotokoll auf die Datenbank spielt. Danach sind die Daten wie beim letzten Systemabsturz. [[7]](#footnote-7)

Auch die besten Transaktionstabellen sind nicht risikofrei. Dort können auch Daten verloren gehen. Selbst das beste RDBMS kann keine hundertprozentige Garantie bieten. [[8]](#footnote-8)

Erfolgreiche RDBMS erfüllen das ACID-Prinzip.

ACID-Prinzip:

* Atomicity – Atomarität
  + Operationen einer Transaktion werden vollständig durchgeführt. Wenn die Transaktion nicht vollständig auf der Festplatte durchgeführt werden konnte, dann wird die ganze Transaktion zurückgesetzt. So können Daten konsistent gehalten werden.
* Consistency – Konsistenz
  + Nicht erlaubte Vorgänge werden nicht in die Datenbank aufgenommen, wie zum Beispiel, wenn ein Benutzer ohne Zugriffsrechte auf die Datenbank zugreifen will. Dieser Vorgang wird sofort abgelehnt.
* Isolation – Isolation
  + Transaktionen können sich gleichzeitig nicht beeinflussen. Das wird durch ein Sperrverfahren praktiziert, indem Daten die gerade bearbeitet werden, gesperrt werden. Mit Teilinformationen wird nicht gearbeitet, da das zu Verwirrungen und Inkonsistenz führen kann.
* Durability – Dauerhaftigkeit
  + Sobald eine Transaktion auf die Datenbank gespeichert wird, steht sie für andere Benutzer zu Verfügung. Wenn ein Fehler auf der Datenbank auftritt, bleibt die Transaktion trotzdem erhalten. (Wenn das keine Hardwarefehler verursachen, die die Festplatte beschädigen können.)[[9]](#footnote-9)

Neben der Vermeidung von Fehlern bei Mehrfachzugriff auf die Datenbank schauen DBMS, dass bei physischen Speicherfehler nicht alle Daten verloren gehen. Es soll statt Totalverlust nur ein kontrollierter Datenverlust vorliegen.[[10]](#footnote-10)

## SQL (Structured Query Language)

SQL ist ein Standard, dass das Kommunizieren mit einer Datenbank unterstützt. Durch SQL Anweisungen kann man Daten aus der Datenbank selektieren, sortieren, erfassen, bearbeiten und mehr. SQL Anweisung fangen mit einem Verb an, wie SELECT und UPDATE, und enden mit einem Semikolon.

## MySQL Datenbank

MySQL ist eine Sprache, dass das kommunizieren mit MySQL Datenbaken unterstützt. Es ist mit den SQL-Standard kompatibel. Die Datenbanksprache unterstützt nicht alle Elemente von SQL, wegen Performance-Gründe. Hingegen stellte sie andere Funktionen zur Verfügung, die in SQL nicht möglich sind. [[11]](#footnote-11) Transaktionen werden bei MySQL nur auf bestimmten Tabellentypen unterstützt.[[12]](#footnote-12)

Folgt …

## MySQL Datenbank im Vergleich zu anderen Datenbanken

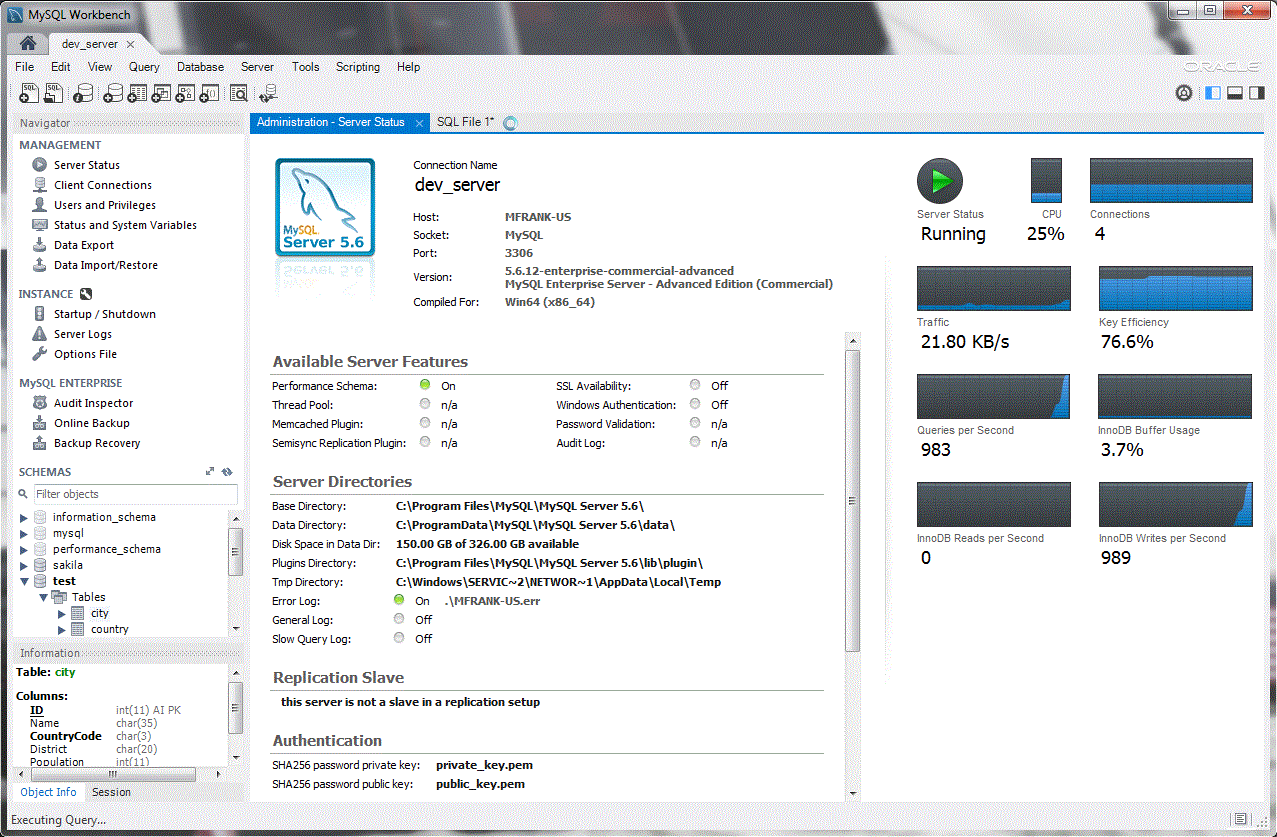
Folgt …

## ER-Modell (Entity-Relationship Modell)

Für die Datenmodellierung soll man das ER-Modell verwenden, damit am Anfang Fehler vermieden werden. Auch wenn Datenbankentwicklern das Erzeugen von Tabellen leicht erscheint, sollen sie dieses strukturiert angehen. Später kann man dadurch Fehler schneller finden und leichter beheben.[[13]](#footnote-13) Wenn man auf die ER-Modellierung verzichtet, muss man sich später nicht nur um die Fehler im Modell kümmern. Man muss sich auch mit Fehlern, in den weitern Programmen, die benützt wurden, wie zum Beispiel in der Rest-Schnittstelle, befassen. Die Suche nach diesen Fehlern ist aufwändiger. Das häufig verwendete ER-Modell hilft Datenbankentwickler aus Informationsstrukturen, Datenstrukturen zu entwickeln. Die Datenstrukturen werden strukturiert abgespeichert, indem Informationsstrukturen erkennt werden und danach miteinander in Beziehung gesetzte werden. Informationsstrukturen entstehen durch die Klassifizierung und Beschreibung von Informationen. Schlussendlich ist das ER-Modell ein Werkzeug, welches mit grafischer Hilfe Tabellenstrukturen erzeugt. [[14]](#footnote-14)

### MySQL Workbench

## MySQL Workbench ist ein vereinheitlichtes visuelles Werkzeug für Datenbankarchitekturen, Entwicklung und Datenbankadministratoren. MySQL Workbench bietet Datenmodellierung, SQL-Entwicklung und weitere Verwaltungstools für Serverkonfiguration, Benutzerverwaltung, Backup etc. MySQL Workbench ist unter Windows, Linux und Mac OS X verfügbar. [[15]](#footnote-15) Das Datenmodell von Jast ist in MySQL Workbench entwickelt worden. Neben der Datenmodellierung wurde die SQL-Entwicklung, Serverkonfiguration und andere Verwaltungstools verwendet.



#### Design

Datenbankadministratoren, Datenbankentwicklung und Datenarchitekten kann man in MySQL Workbench visuell entwerfen, modellieren, generieren und verwalten. ES umfasst alles, was ein Datenmodellierer zum Erstellen von komplexen ER-Modellen benötigt. Außerdem können visuelle Datenbankdesigns mittel Forward- Engineering zu Tabellen umgewandelt werden und umgekehrt mittels Reverse-Engineering. Durch diesen Vorgang kann man bei einer Änderung Zeit sparen.[[16]](#footnote-16)

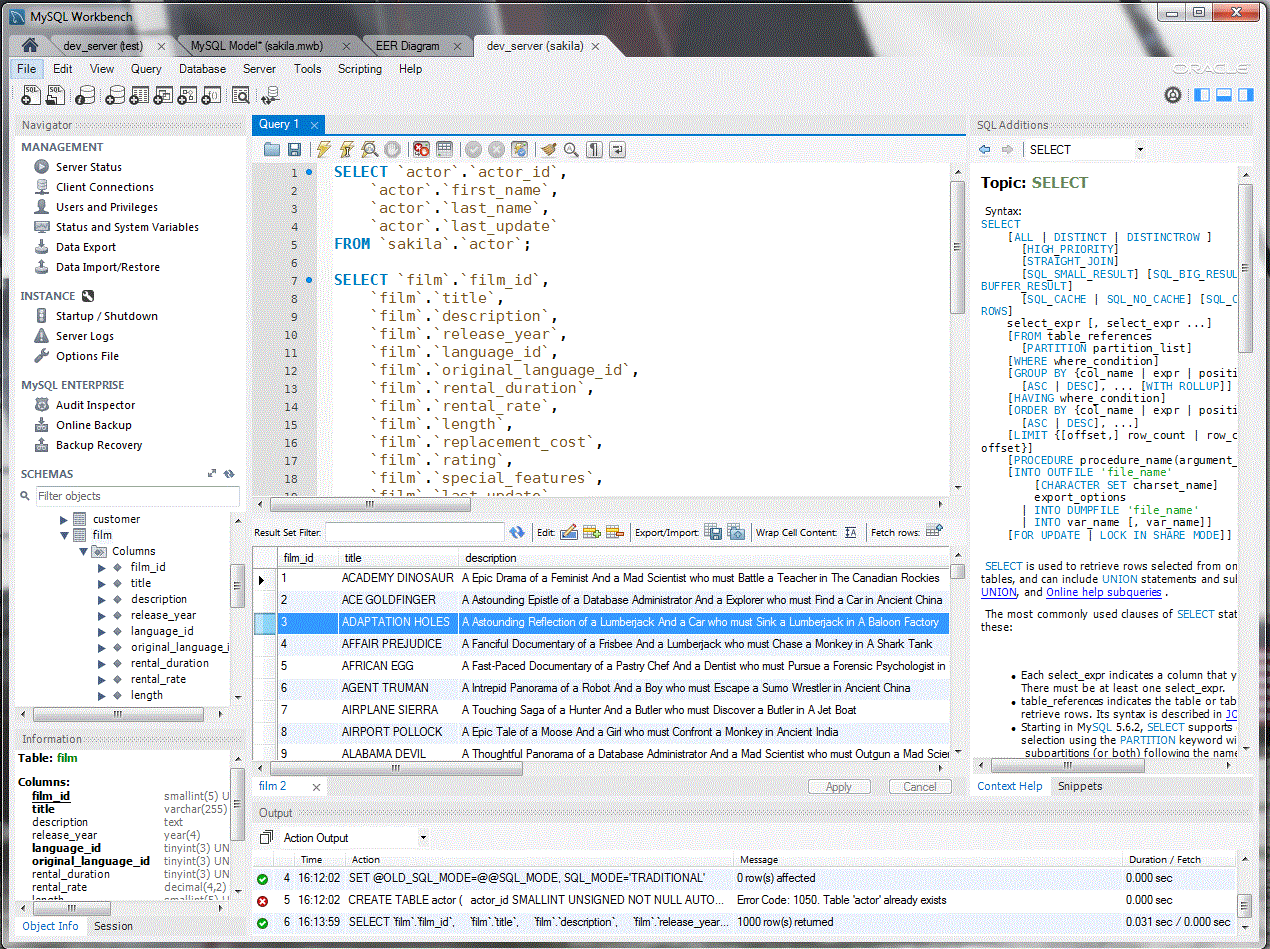
#### Entwicklung

In MySQl Workbench können SQL-Abfragen durch visuelle Tools zum Erstellen, zum Ausführen und zum Optimieren dargestellt werden. Außerdem können bestimmte Entitäten durch Farben hervorgehoben, automatisch vervollständigt und durch kopieren und einfügen wiederverwendet werden. Das Datenbankverbindungsfenster ermöglicht Entwicklern die einfache Verwaltung von Standard-Datenbankverbindungen, einschließlich MySQL Fabric. Es bietet außerdem sofortigen Zugriff auf Datenbankschema und Objekte. [[17]](#footnote-17)



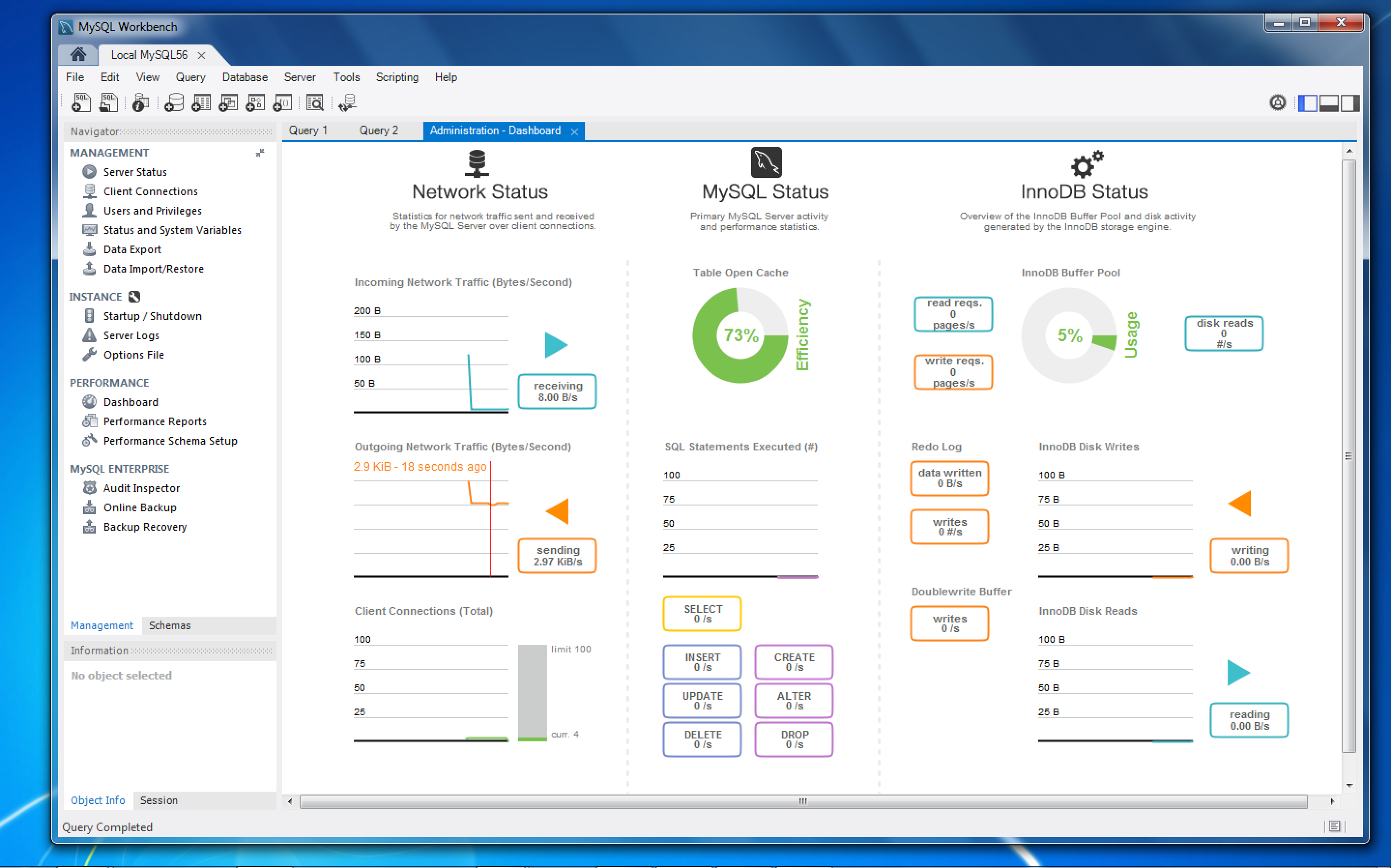
#### Verwaltung

MySQL Workbench bietet eine visuelle Konsole zur einfachen Verwaltung von MySQL-Umgebungen und zur besseren Übersicht der Datenbanken. Entwickler und Datenbankadministratoren können die visuellen Tools zum Konfigurieren von Servern, Verwalten von Benutzern, Durchführen von Sicherungen und Wiederherstellungen, Überprüfen von Überwachungsdaten und Anzeigen des Datenbankzustands verwenden. [[18]](#footnote-18)



#### Visuelles Leistungs-Dashboard

Mithilfe von Leistungsdashboards können wichtige Leistungsindikatoren der Datenbank angezeigt werden. Leistungsberichte bieten eine einfache Identifizierung und Zugriff auf IO-Hotspots, kostenintensive SQL-Anweisungen und mehr. Durch verschiedenen Tools von MySQL-Anwendungen kann man dann die Leistung verbessern. Durch einen Klick können Entwickler sehen welche Befehle einen Verbesserungsbedarf haben und sie können den Visual Explain Plan besichtigen.



#### Datenbankmigration

MySQL Workbench bietet jetzt eine vollständige, benutzerfreundliche Lösung für die Migration von Microsoft SQL Server-, Microsoft Access-, Sybase ASE-, PostreSQL- und anderen RDBMS-Tabellen, -Objekten und -Daten nach MySQL zu umwandeln. Die Umwandlung kann von Entwickler und Datenbankadministratoren, durch vorhandene Anwendungen, schnell und einfach unter MySQL, auf Windows und anderen Plattformen, erfolgen. Migration unterstützt auch frühere Versionen von MySQL.

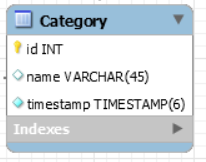
### Wichtige Objekte der Datenmodellierung

#### Entity

Eine Entität ist ein strukturiertes Datenobjekt mit Eigenschaften (Attributen). Die Attribute besitzen jeweils einen definierten Wertebereich. Einem Attribut können keine Listen und Mengen zugewiesen werden.[[19]](#footnote-19)

##### Entity-Typen

Die Vereinigung der Entitäten mit gleichen Eigenschaften, wie alle Personen in einer Tabelle, wird als Entity-Typ bezeichnet. Entitäten die ohne andere Entitäten nicht vom Wert sind, wie Adresse bei Personal, werden in sogenannte schwache Entity-Typen vereinigt. Ein oder mehrere Attribute müssen als Primärschlüssel definiert werden, um die Entitäten eines Entity-Typen eindeutig zu machen. Entitätstypen werden in im ER-Modell als Rechteck dargestellt und die Eigenschaften werden meistens als Ovale mit dem Entitätstyp verbunden. [[20]](#footnote-20) In MySQL Workbench stehen die Attribute im Entitätstyp.

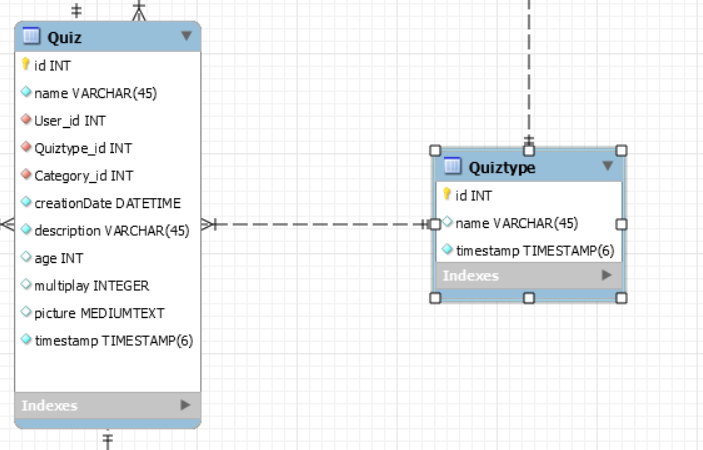


#### Relationships (Beziehungen)

Es werden zwischen mindestens zwei Entitäten Beziehungen gesetzt. Beziehungen sind keine Entitäten. Siehe Abbildung …

##### Relationships-Typen

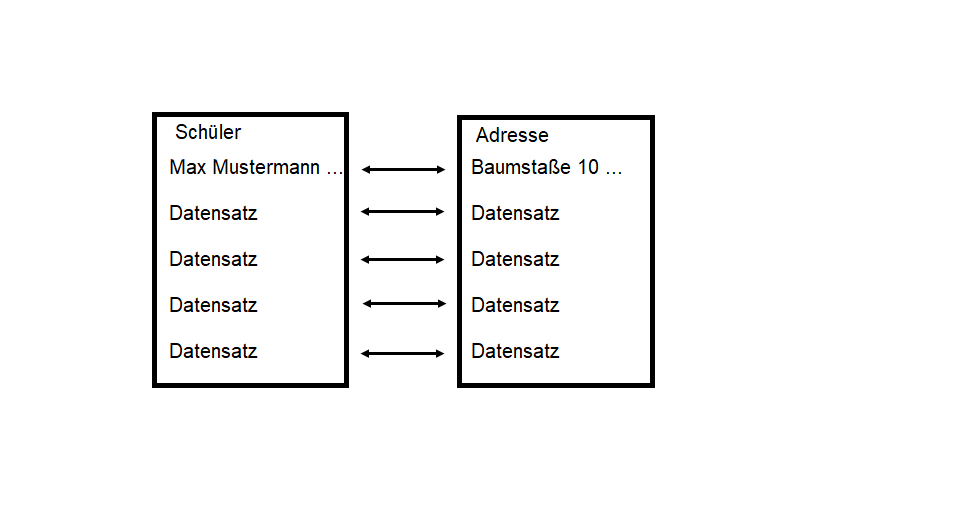
Relationstypen können auch Attribute besetzen, die eine Beziehung genauer beschreibt. Sie werden meisten als Raute dargestellt. In MySQL kann man Attribute nur in einem Kommentar bekannt geben.



##### 

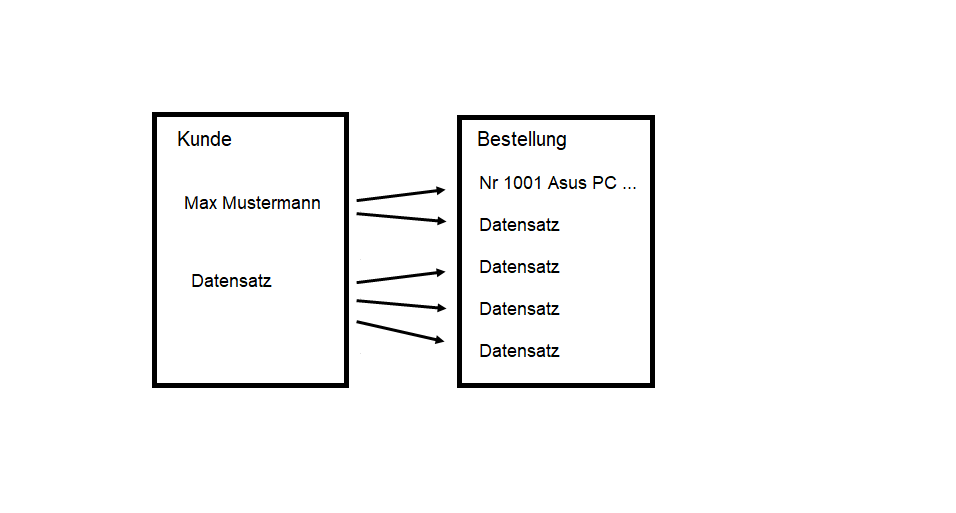
##### Eins zu eins (1:1)

Eine Eins-zu-eins-Beziehung verbindet zwei Entitäten die sich gegenseitig erfordern. Bei einer Eins-zu-eins-Beziehung ist ein Eintrag aus der ersten Tabelle mit genau einen Eintrag der anderen Tabelle zugeordnet. [[21]](#footnote-21) Zum Beispiel, wenn in einer Tabelle die Stammdaten der Schüler sind und in der zweiten Tabelle die Adressen der Schüler.



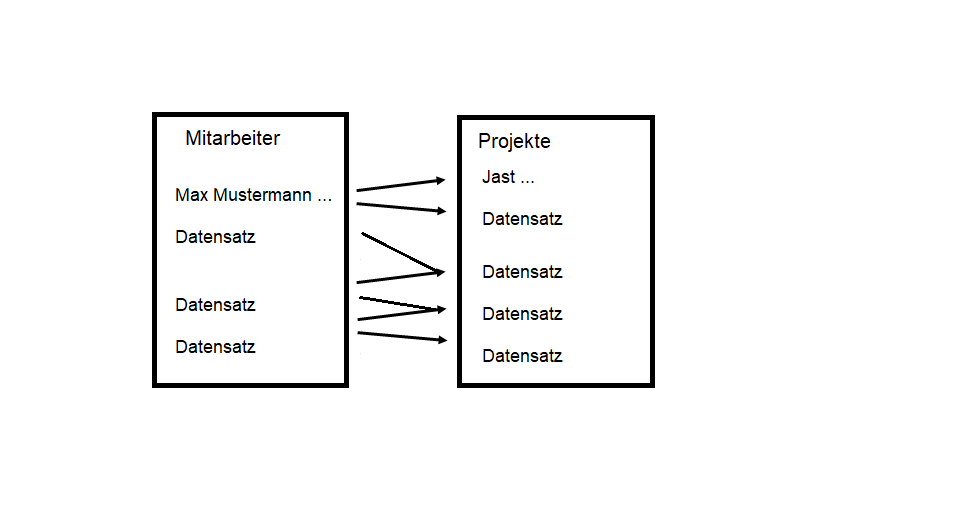
##### Eins zu vielen (1:n)

Bei einer Eine-zu-vielen-Beziehung steht eine Entität eines Entity-Typen mir mehreren Entitäten eines anderen Entity-Typen in Beziehung. [[22]](#footnote-22) Zum Beispiel, wenn in einer Tabelle die Kunden aufgelistet sind und in der anderen die Bestellungen. Die Kunden können mehrere Bestellungen haben, somit ist das eine Eine-zu-vielen-Beziehung.



##### Viele zu vielen (n:m)

Bei einer Vielen-zu-vielen-Beziehung können mehrere Entitäten eines Entity-Typen zu mehreren Entitäten eines anderen Entity-Typen in Beziehung stehen. Eine n:m Beziehung wird in der Praxis oft benutzt. Eine n:m Beziehung wird mithilfe einer Zwischentabelle in die Datenbank gespeichert. Die n:m Beziehung wird zu 2 1:n Beziehungen zerlegt. [[23]](#footnote-23) Zum Beispiel, wenn in einer Tabelle die Mitarbeiter aufgelistet sind und in der anderen Tabelle die Projekte. Die Mitarbeiter können an mehrere Projekte arbeiten, somit ist das eine Vielen-zu-vielen-Beziehung.



### Das relationale Datenmodell

Das weitverbreitete Datenmodell dient als konzeptionelle Grundlage von rationalen Datenbanken. Hier werden Entity-Typen und Relationship-Typen als Relationen bezeichnet und in Tabellenform gespeichert. Eine Relation kann immer als Tabelle dargestellt werden, wo die Zeilen Datensätze und die Spalten Eigenschaften entsprechen. Zum Abfragen von Daten wird die standardisierte Sprache SQL verwendet. Durch SQL können Daten abgefragt, verknüpft, erweitert oder gelöscht werden. Eigentlich dient SQL nur zum Abfragen von Daten. Bei den weiteren Funktionen spricht man auch von der DML (Data Manipulation Language). Die Daten müssen mindestens in der ersten Normalform sein um gespeichert werden zu können. [[24]](#footnote-24)

### Vom ER-Modell zum rationalen Datenmodell

Zuerst werden aus den Entitäten und den Beziehungen aus dem grafischen ER-Modell Tabellen gebildet. Danach werden Datentypen der Eigenschaften und die Primärschlüssel der Tabelle bestimmt. [[25]](#footnote-25)

#### Hier müssen folgendes beachtet werden:

**Bei starken Entity-Typen**

Nicht schwache Entitäten werden direkt zur Tabelle. Die erste Spalte in der Tabelle beinhaltet den Primärschlüssel. Der Primärschlüssel kann aus mehreren Spalten zusammengesetzt werden. Diese dienen zur eindeutigen Identifizierung. Nummerische Primärschlüssel sind effizient.[[26]](#footnote-26)

**Schreibweise**

*Bestellung(****BNr*** *number, Bezeichnung varchar2(20))*

**Bei -1: n Beziehungen**

In der Primärtablelle (n-Relation) wird eine Spalte mit dem Primärschlüssel der Sekundärtabelle (1-Relation) gesetzt. In der Primärtabelle ist der Primärschlüssel der Sekundärtabelle ein Fremdschlüssel. [[27]](#footnote-27)

**Bei 1:1-Beziehungen**

Die 1:1-Beziehung ist ein Spezialfall der 1:n Beziehung, somit wird es wie bei der 1:n Beziehung gespeichert. Der Unterschied hier ist das man die freie Auswahl hat, welche Entität die Primärtablelle und welche Entität die zur Sekundärtabelle wird. [[28]](#footnote-28)

**Schreibweise**

*Bestellung (****BNr*** *number, Bezeichnung varchar2(20))*

*Kunde (****KNr*** *number, BNr number, Vorname varchar2(20), Nachname varchar2(20), GebDatum date)*

**Bei n:m-Beziehungen**

Es wird eine Zwischentabelle erstellt, die von beiden Tabellen den Primärschlüssel enthält. In der Zwischentabelle sind beide Schlüssel Primär- und Fremdschlüssel.Die Zwischentabelle kann auch weitere Eigenschaften enthalten. [[29]](#footnote-29)

**Schreibweise**

*Bestellung (****BNr*** *number, Bezeichnung varchar2(20))*

*Gruppenbestellung (****BNr*** *number,,****KNr*** *number)*

*Kunde (****KNr*** *number, Vorname varchar2(20), Nachname varchar2(20), GebDatum date)*

**Bei schwachen Entity-Typen**

Schwache Entitäten werden wie starke Entitäten gebildet. Zusätzlich wird der Primärschlüssel der staken Entität in der Tabelle der schwachen Entität als Fremdschlüssel eingetragen, dieser gilt dann als weiterer Primärschlüssel. [[30]](#footnote-30)

**Schreibweise**

*Adresse (****ANr*** *number, Straße varchar2(20), Hausnummner)*

*Ort (****ONr*** *number,,ANr number,PLZ)*

### Normalformen von Relationen

**Erste Normalform**

**Zweite Normalform**

**Dritte Normalform**

Datenmodellierung anhand von Jast

Hier werden die Regeln des ER-Modells schrittweise anhand des Jast ER-Modell erklärt:

## Sicherung der Datenbank

Folgt …

Buch: Jetzt lerne ich PHP 5 & MySQL 4.1 ISBN: 3-8272-6534-7

<https://www.amazon.de/Jetzt-lerne-MySQL-objektorientierte-Webprogrammierung/dp/3827265347/ref=sr_1_1?ie=UTF8&qid=1519186184&sr=8-1&keywords=3827265347&dpID=51oMBBj6fGL&preST=_SY264_BO1,204,203,200_QL40_&dpSrc=srch>

<http://www.searchenterprisesoftware.de/definition/Datenbank-Managementsystem-DBMS>

1. (**vgl**. LETZEL & BETZ 2005, S.208)  [↑](#footnote-ref-1)
2. (**vgl**. Suehring 2002, S.105)  [↑](#footnote-ref-2)
3. (**vgl**. LETZEL & BETZ 2005, S.28 f.)  [↑](#footnote-ref-3)
4. (**vgl**. Suehring 2002, S.105)  [↑](#footnote-ref-4)
5. (**vgl**. LETZEL & BETZ 2005, S.208)  [↑](#footnote-ref-5)
6. (**vgl**. LETZEL & BETZ 2005, S.358)  [↑](#footnote-ref-6)
7. (**vgl**. Suehring 2002, S.106 f.)  [↑](#footnote-ref-7)
8. (**vgl**. Suehring 2002, S.106 f.)  [↑](#footnote-ref-8)
9. (**vgl**. Suehring 2002, S.106 f.)  [↑](#footnote-ref-9)
10. (**vgl**. LETZEL & BETZ 2005, S.239 f.)  [↑](#footnote-ref-10)
11. (**vgl**. LETZEL & BETZ 2005, S.209)  [↑](#footnote-ref-11)
12. (**vgl**. LETZEL & BETZ 2005, S.209)  [↑](#footnote-ref-12)
13. (**vgl**. LETZEL & BETZ 2005, S.209)  [↑](#footnote-ref-13)
14. (**vgl**. LETZEL & BETZ 2005, S.209)  [↑](#footnote-ref-14)
15. (https://www.mysql.com/de/products/workbench/) [↑](#footnote-ref-15)
16. (https://www.mysql.com/de/products/workbench/) [↑](#footnote-ref-16)
17. (https://www.mysql.com/de/products/workbench/) [↑](#footnote-ref-17)
18. (https://www.mysql.com/de/products/workbench/) [↑](#footnote-ref-18)
19. (**vgl**. LETZEL & BETZ 2005, S.209)  [↑](#footnote-ref-19)
20. (**vgl**. LETZEL & BETZ 2005, S.210)  [↑](#footnote-ref-20)
21. (**vgl**. LETZEL & BETZ 2005, S.211 f.)  [↑](#footnote-ref-21)
22. (**vgl**. LETZEL & BETZ 2005, S.212)  [↑](#footnote-ref-22)
23. (**vgl**. LETZEL & BETZ 2005, S.213)  [↑](#footnote-ref-23)
24. (**vgl**. LETZEL & BETZ 2005, S.213 f.)  [↑](#footnote-ref-24)
25. (**vgl**. LETZEL & BETZ 2005, S.214)  [↑](#footnote-ref-25)
26. (**vgl**. LETZEL & BETZ 2005, S.214)  [↑](#footnote-ref-26)
27. (**vgl**. LETZEL & BETZ 2005, S.215)  [↑](#footnote-ref-27)
28. (**vgl**. LETZEL & BETZ 2005, S.215)  [↑](#footnote-ref-28)
29. (**vgl**. LETZEL & BETZ 2005, S.216)  [↑](#footnote-ref-29)
30. (**vgl**. LETZEL & BETZ 2005, S.216)  [↑](#footnote-ref-30)