

Datenbanken

Reinhard Nitzsche
Version 0.3 (05.02.2020)

1 Vorwort

Julian Dorns InstaHub ist ein wundervolles Werkzeug, um Datenbanken zu unterrichten. Das vorliegende Skript orientiert sich an dem von ihm vorgeschlagene Vorgehen, weicht aber an einigen Stellen ab.

Zielgruppe des Skriptes und des unterstützenden Moodle-Kurses sind Schüler*innen mit Mittlerem Bildungsabschluss auf dem Weg zur (Fach-)Hochschulreife. Entstanden ist es für den Unterricht in beruflichen Vollzeitschularten. Der Einsatz in allgemeinbildenden Schulen und mit leistungsstarken jüngeren Schüler*innen ist aber möglich.

1.1 Pfade durch dieses Skript

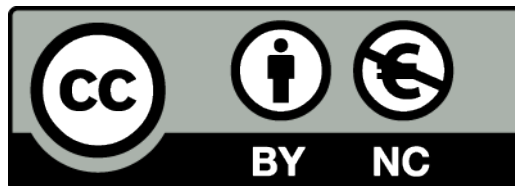
Einige Kapitel sind optional. Sie behandeln interessante Themen, die aber in den späteren Kapiteln nicht wieder aufgegriffen werden. Sie können also weggelassen werden, wenn dies notwendig ist. Eventuell ist auch ein Einsatz im Rahmen der Binnendifferenzierung sinnvoll.

Dies folgenden Kapitel sind **nicht optional**:

Nr.	Kapitel	Bemerkungen
1	Einstimmung	Auf die Eigenschaften von DBS wird zwar im weiteren Verlauf zurückgegriffen, jedoch in geringem Umfang. Daher kann auf Kapitel 1 verzichtet werde, wenn auch ungerne.
2	Erste Schritte	
3	Einfache SQL-Abfragen	
4	SQL-Abfragen über mehr als eine Tabelle	
5		
17	Top 10 der Themen, die wir nicht behandelt haben	

Reinhard Nitzsche

1.2 Lizenz



Logo Lizenz "Creative Commons Namensnennung - Nicht kommerziell - Weitergabe unter gleichen Bedingungen"

Datenbanken. Eine Einführung mit Instahub von Reinhard Nitzsche ist lizenziert unter einer [Creative Commons Namensnennung-Nicht kommerziell 3.0 Deutschland Lizenz](#).

Klarstellung: Der Einsatz im Unterricht und in Arbeitsgemeinschaften in öffentlichen Schulen ist nicht kommerziell.

1.3 Nur möglich durch

Dieses Skript entstand unter Einsatz folgender Software:

1. Die Markdown-Dateien wurden mit Typora editiert, die
2. pandoc in verschiedene Zielformate übersetzt hat, darunter
3. ODT, aus dem Libre Office und PDFCreator dann PDF-Dateien erzeugt haben.
4. Die Screenshots wurden mit Greenshot erzeugt und annotiert.
5. Der Moodle-Kurs wurde wäre ohne Moodle undenkbar (ach nee...),
6. Die Lernspiele im Moodle-Kurs entstanden bei LearningApps.com.
7. Die Screencasts im Moodle-Kurs wurden mit OBS erstellt.

Allen an der Entwicklung dieses Tools beteiligten herzlichen Dank.

Wertvolle Hinweise verdanke ich den Schüler*innen der Klassen Abi21b, BOM19-2 und BOM19-3 der Beruflichen Schule Elmshorn.

1.4 Bildnachweis

Alle verwendeten Grafiken sind eigene Werke oder stammen aus der Public Domain:

- tango-Projekt
- Open Clipart Library

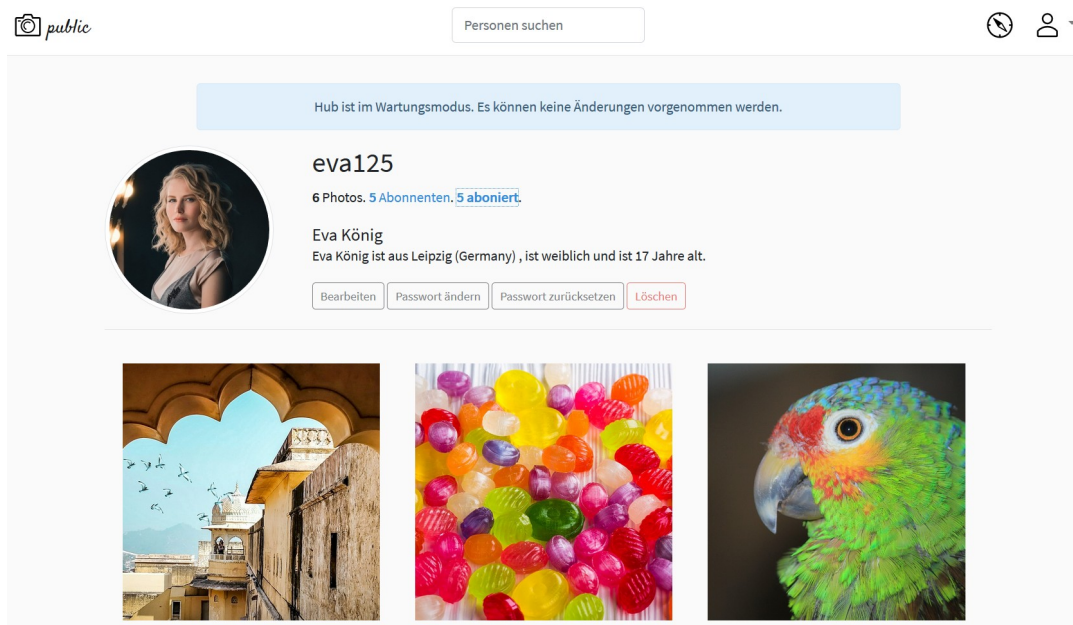
Allen Zuträger*innen herzlichen Dank.

2 Einstimmung



2.1 InstaHub?

InstaHub ist ein soziales Netz, das Instagram ähnelt. Es gibt aber einen ganz wichtigen Unterschied: Anders als Instagram lässt InstaHub Sie hinter die Kulissen blicken. So gewinnen Sie Einblicke in die Arbeitsweise von sozialen Netzen und lernen ganz nebenbei, wie Datenbanken funktionieren.

Die folgenden Screenshots geben Ihnen einen Eindruck davon, was Sie in InstaHub erwartet:








In InstaHub können Sie Fotos hochladen.



public  

Hub ist im Wartungsmodus. Es können keine Änderungen vorgenommen werden.

Abonnenten von eva125






	maria422 Maria Baumgartner 5 Photos. 8 Abonnenten. 17 aboniert.	Folgen
	maja260 Maja Wulf 9 Photos. 6 Abonnenten. 12 aboniert.	Folgen
	xenia437 Xenia Freud 4 Photos. 6 Abonnenten. 13 aboniert.	Folgen
	bea414 Bea Beich 14 Photos. 10 Abonnenten. 10 aboniert.	Folgen
	lennox87 Lennox Reinhardt 8 Photos. 6 Abonnenten. 15 aboniert.	Folgen

*Sie können Ihren Freund*innen folgen ...*

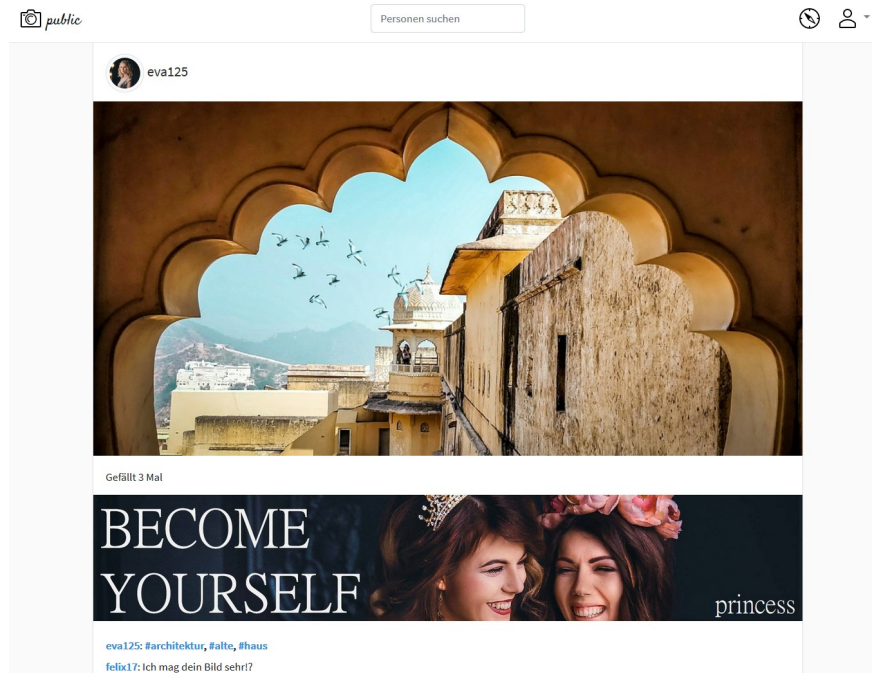
public  

Hub ist im Wartungsmodus. Es können keine Änderungen vorgenommen werden.

Personen die eva125 folgen

	leonard146 Leonard Schwartz 7 Photos. 6 Abonnenten. 7 aboniert.	Entfolgen
	aaron183 Aaron Lange 12 Photos. 7 Abonnenten. 10 aboniert.	Entfolgen
	josia137 Josia Schwab 3 Photos. 8 Abonnenten. 8 aboniert.	Entfolgen
	lena469 Lena Weiß 13 Photos. 8 Abonnenten. 7 aboniert.	Entfolgen
	marlo152 Lena Krause 6 Photos. 9 Abonnenten. 8 aboniert.	Entfolgen

*... und Ihre Freund*innen können Ihnen folgen!*



Fotos können kommentiert und bewertet werden.

InstaHub funktioniert also fast wie Instagramm. Inklusive Werbung. Der entscheidende Unterschied ist, dass alles in InstaHub fiktiv ist. Die Menschen gibt es genauso wenig wie die Unternehmen, die die Werbung schalten.

Unter <https://instahub.org/#guest> können Sie sich schon einmal umsehen.

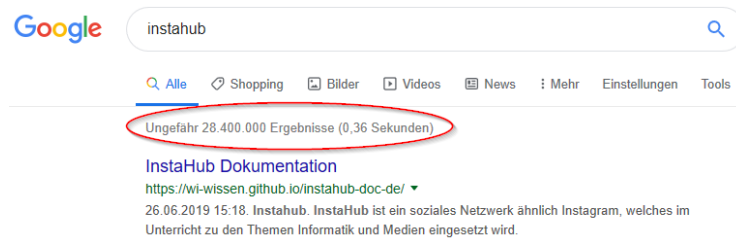
2.2 Datenbanken sind überall

Programme, die Datenbanken verwalten, heißen **Datenbankmanagementsysteme** (DBMS). Kleinere DBMS sind Microsoft Access oder Libre Office Base. Größere DBMS sind MySQL und der MS SQL-Server.

Im Grund stellen die meisten DBMS lediglich die Möglichkeit bereit, gigantische Tabellen zu erstellen und sie in Windeseile zu verknüpfen.

Wenn Sie zum Beispiel mit Google¹ nach "instahub" suchen, schlägt Google in seinen Datenbanken nach und stellt in 0,36 s fest, dass sie fast 30 Millionen Treffer hat:

¹ Die anderen Suchmaschinen machen es genauso, aber die Zeit gibt nur Google aus...



Google findet in unter einer Sekunde fast 30 Millionen Websites zum Thema “Instahub”!

Kein Onlineshop kommt ohne Datenbanken aus - z. B. für die Produktpräsentation und die Verwaltung der Bestellungen:



Onlineshops sind ohne Datenbanken nicht vorstellbar (Quelle: www.buecher.de)

Aber auch beim Versand der Ware geht nichts ohne Datenbanken:

Sendungsstatus			
Überblick		Detailsansicht	
		Datum	Standort
✓	Zugestellt	03.07.2019 9:52	ELMSHORN, DE
✓	Zustellung läuft	03.07.2019 7:26	Hamburg, Germany
✓	Versendet	02.07.2019 20:55	Frankfurt, Germany
✓	Aufkleber erstellt	02.07.2019 17:01	Germany

Jedes Paket hinterlässt seine Spuren in Datenbanken (Quelle: UPS.COM)

Aufgabe 2.1: Datenbanken sind überall

Überlegen Sie: Wo sind Sie in den letzten Tagen wohl Datenbanken begegnet? Wo vermuten Sie Datenbanken, die Informationen über Sie speichern?

2.3 Eigenschaften von DBMS

2.3.1 Warum Datenbanken, wir haben doch Tabellenkalkulationen!

Wenn die meisten Datenbanken aus lauter Tabellen bestehen, drängt sich die Frage auf, warum wir dafür nicht einfach eine Tabellenkalkulation wie z. B. Microsoft EXCEL oder Libre Office Calc verwenden. Hier also die wichtigsten Gründe für die Verwendung von Datenbankmanagementsystemen (DBMS):

2.3.2 Viele Köche verderben nicht immer den Brei

DBMS ermöglichen in der Regel zahlreichen Benutzern gleichzeitig den Zugriff auf die gespeicherten Daten. Man spricht auch von **Mehrbenutzerfähigkeit**.

Mehrere Benutzer können also zeitgleich auf eine Tabelle zugreifen, eventuell sogar auf die selbe Zeile (Bei Datenbanken spricht man von *Datensatz*). Tabellenkalkulationen erlauben dies in der Regel allenfalls zum lesen.

2.3.3 Was ich darf, darfst Du noch lange nicht!

DBMS erlauben **Zugriffsbeschränkungen**. Das heißt, dass nicht alle Benutzer*innen über die selben Rechte verfügen. Einige typische Rechte sind z. B.:

- Erstellen neuer Tabellen
- Ändern der Struktur bestehender Tabellen
- Datensätze ansehen
- Datensätze einfügen
- Datensätze löschen
- Datensätze verändern

2.3.4 Jedem das Seine

DBMS müssen den unterschiedlichen Bedürfnissen und Fähigkeiten der Benutzer *innen Rechnung tragen und für jede Benutzergruppe unterschiedliche Schnittstellen zur Verfügung stellen. Man spricht von **Mehrbenutzerschnittstellen**. Z. B. gibt es Oberflächen für gelegentliche Nutzer*innen, oder Schnittstellen für die Programmierer*innen von Datenbank Anwendungen oder hochentwickelte Anfragesprachen für versierte Nutzer*innen mit sehr speziellen Informationsbedürfnissen.

2.3.5 Ich sehe was, was Du nicht siehst!

DBMS ermöglichen die Einrichtung von Datensichten (*views*), d. h. unterschiedlicher Perspektiven auf die Datenbank. So könnte z. B. jede*r Dozent*in einer Universität eine Liste aller Teilnehmer*innen seines Kurses inklusive der vorher besuchten Kurse sehen, obwohl diese Daten in vielen Tabellen verstreut liegen.

2.3.6 Lass mich mal abschreiben!

Wie viele Stellen im Betrieb benötigen eigentlich die Adresse eine*r Kund*in? Mindestens folgende:

- Kundenbuchhaltung,

- Versand und
- Marketing.

Dieses mehrfache Speichern derselben Sachverhalte nennt man Redundanz. (lat. *redundare* „im Überfluss vorhanden sein“). Redundanz bringt Probleme mit sich:

- Wenn ein- und dieselben Daten an verschiedenen Stellen gespeichert sind, werden Daten mehrfach erfasst (mehrfacher Zeitaufwand) und mehrfach gespeichert (mehrfacher Speicherplatz).
- Das wäre noch nicht so schlimm. Es gibt noch ein viel größeres Problem: Das der Inkonsistenz. (lat.: *in* nicht, *con* zusammen, *sistere* halten) Die Wahrscheinlichkeit ist groß, dass irgendwann Versand und die Kundenbuchhaltung unterschiedliche Adressen verwenden, weil der Kunde umgezogen ist, aber einer von beiden das nicht mitbekommen hat. Die Pakete gehen dann z. B. schon an die neue Adresse, während die Rechnung an die alte Adresse geht.

DBMS sollten über eine **Redundanzkontrolle** verfügen, so dass bei (manchmal unvermeidlicher) redundanter Speicherung von Daten Inkonsistenzen verhindert werden.

2.3.7 Beziehungskisten

Die Tabellen einer Datenbank hängen untereinander logisch zusammen. DBMS unterstützen diese komplexen Beziehungen, in dem sie diese Beziehungen verwalten und effizient miteinander verbinden und aktualisieren können. Z. B. dient die Personalnummer in vielen Tabellen der Personalabteilung zur Identifizierung eines Mitarbeiters. Diese **Beziehung zwischen Daten** nennt man auch **Relation**. Sie ist so wichtig, dass die wichtigste Gruppe von DBMS sogar nach ihr benannt wurde: Die **relationalen Datenbanken**. Instahub liegt eine solche relationale Datenbank zu Grunde. Auch Access und Base sind relationale Datenbanken.

2.3.8 Vertrauen ist gut, Kontrolle besser!

Datenbanken können in DBMS so definiert werden, dass bestimmte **Integritätsbedingungen** (Integrität: lat. *integritas* „unversehrt“, „vollständig“) zu jedem Zeitpunkt eingehalten sein müssen. Auf diese Weise kann z. B. sicher gestellt werden, dass jeder Kurs eine*n Dozent*in bekommt und kein*e Student*in einen Kurs belegen kann, den es gar nicht gibt.

2.3.9 Daten, Daten, Daten – soweit das Auge reicht!

DBMS sind darauf ausgerichtet, mit sehr großen Datenmengen umgehen zu können. Das Selektieren von tausenden Datensätzen aus einer Datenbank

mit Millionen von Datensätzen dauert in modernen DBMS nur Bruchteile von Sekunden.

2.3.10 Sicher ist sicher!

Sollte es zu einem Problem mit dem DBMS kommen, z. B. wegen eines Soft- oder Hardwarefehlers, sorgen DBMS durch geschickte Speicherung der Daten auf der Festplatte für die Möglichkeit, auf einem korrekten Zustand kurz vor Entstehen des Problems wieder aufzusetzen. (**Recovery**)

Aufgabe 2.2: Eigenschaften von DBMS

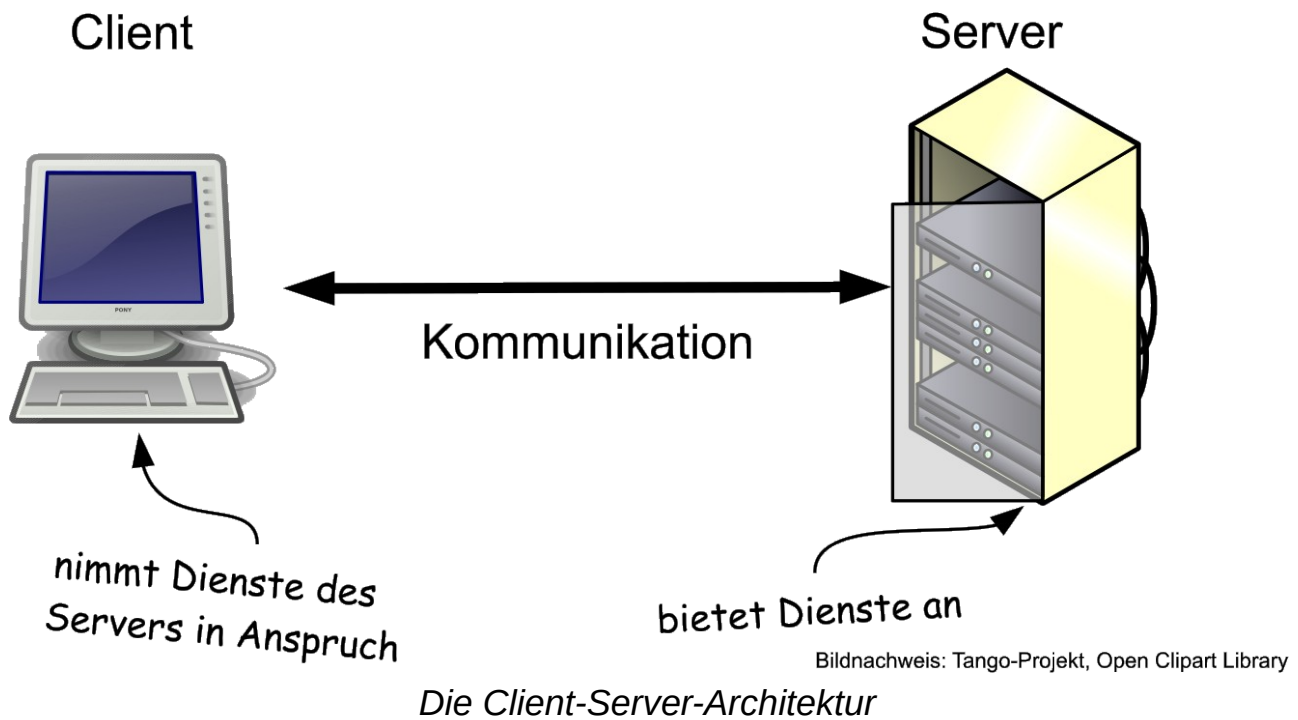
Erstellen Sie eine Tabelle, in der Sie zu jeder in diesem Abschnitt genannten Eigenschaft von DBMS ein Beispiel aus einem Warenwirtschaftssystem eines Unternehmens finden. Warenwirtschaftssysteme dienen zur Abbildung der Warenströme in Unternehmen.

2.4 Die Client-Server-Architektur

Es vermeidet Missverständnisse, wenn wir vor der Arbeit mit InstaHub noch einen Blick auf die Client-Server-Architektur (oft auch: Client-Server-Modell) werfen.

Den Begriff “Server” kennen sie bestimmt bereits. Zum Beispiel wissen Sie, dass ein Server ausfallen kann und dann etwas nicht mehr funktioniert, auf das Sie dringend angewiesen sind.

Beispiel: Herr Ölgemöller arbeitet in der Buchhaltung der Hausverwaltung Sörkeling. In seinem Büro befindet sich ein Drucker, auf den er über das lokale Netz (LAN) zugreift. Der Druckserver ist ausgefallen. Obwohl sowohl sein Rechner als auch der Drucker dienstbereit sind, kann Herr Ölgemöller nicht drucken, da der vom Server angebotene Dienst “Drucken” nicht verfügbar ist.



Bei der Client-Server-Architektur gibt es mindestens zwei IT-Systeme. Das eine ist der Server, der Dienste anbietet und das andere ist der Client, der diese Dienste in Anspruch nehmen kann. Meist greifen mehrere Clients auf einen einzigen Server zu.

In einem IT-Lexikon liest sich das dann so:

Client-Server-Architektur (engl.: *client-server architecture*) Kooperati-
ve Form der Informationsverarbeitung bei der sich ergänzende Soft-
warekomponenten meist (aber nicht zwangsläufig) auf unterschiedli-
che Rechner verteilt werden, die über ein Rechnernetz verbunden
sind. Ein Client-Server-System ist eine einfache Form eines verteilten
Systems, bei dem die Rollenverteilung zwischen dem Serverprogramm
und Klientenprogramm statisch ist und meist eine 1:n-Beziehung dar-
stellt. (Vgl. Hansen, S. 64)

Beachten Sie: Client und Server sind im Sinne der Client-Server-Architektur *Softwarekomponenten* und keine *Hardware* (Geräte)! Daher können Client und Server auch durchaus auf ein- und demselben Rechner laufen. Z. B. wird bei der Entwicklung von Websites gerne auf das Paket XAMP zurück gegriffen. Es stellt einen vollen Webserver zur Verfügung, der u.a. aus den folgenden Komponenten besteht:

- HTTP-Server
- Datenbankserver
- Mailserver

Wenn es InstaHub nicht gebe, wäre der Einsatz von XAMP eine Möglichkeit, im Unterricht mit Datenbanken zu arbeiten. Wir arbeiten aber mit InstaHub.

Jede*r Schüler*in wird im Browser (=Client) die Website von Instahub aufrufen und dort Datenbankabfragen an den Server senden. Sie brauchen den Inhalt Ihres Instahubs also nicht auf einem USB-Stick zu speichern. Ihre Daten sind immer auf dem Server. Alles was Sie brauchen, ist ein Internetfähiger Rechner und eine Internetverbindung.

2.5 Quellen zu diesem Kapitel

- Abschnitt [Eigenschaften von DBMS](#):
 - *Elmasri*, Rami und Shamkant B. Navathe: Grundlagen von Datenbanksystemen. Ausgabe Grundstudium. 3. Auflage. Pearson. S. 24-35.
- Abschnitt [Die Client-Server-Architektur](#):
 - *Hansen*, H. R. und Neumann, G.: Arbeitsbuch Wirtschaftsinformatik. IT-Lexikon, Aufgaben, Lösungen. UTB. 7. Auflage.

3 Erste Schritte mit InstaHub

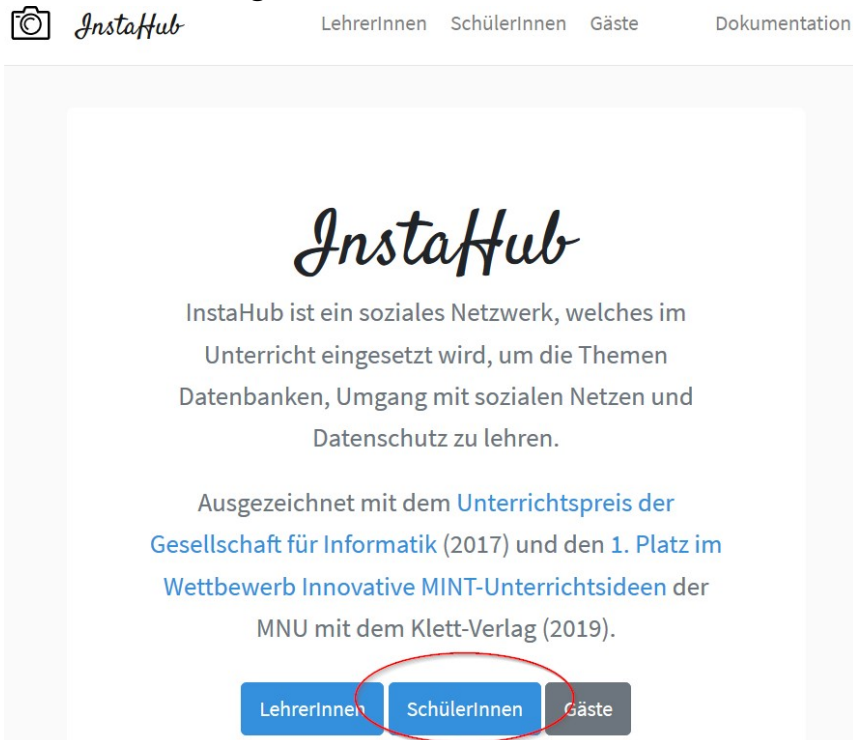
3.1 Den eigenen Hub erstellen

Aufgabe 3.1: Beantragen Sie einen eigenen InstaHub!

Zur Arbeit mit InstaHub arbeitet jede*r Schüler*in auf einem eigenen InstaHub. Es hat also jeder sein eigenes soziales Netzwerk.

Um einen InstaHub zu beantragen, gehen Sie wie folgt vor:

1. *Rufen* Sie InstaHub in Ihrem Browser (**nicht** Internet Explorer!)² auf: <https://instahub.org/>!
2. *Klicken* Sie unten mittig auf den Button SchülerInnen:

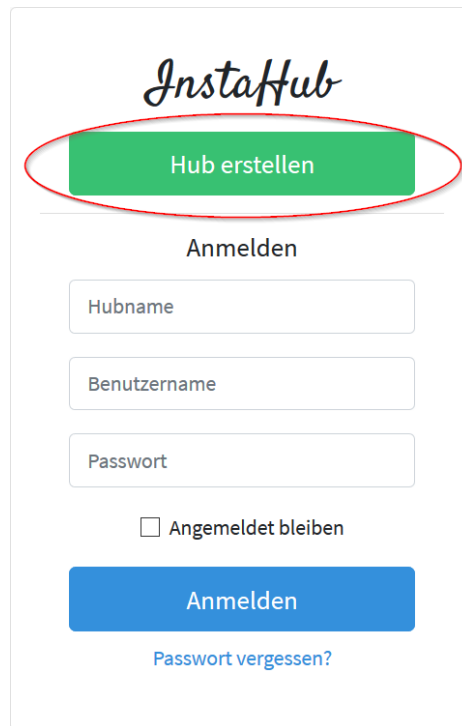


Einen eigenen InstaHub anlegen - Schritt 1

3. Klicken Sie auf Hub erstellen:

² InstaHub unterstützt alle gängigen Browser mit Ausnahme des Internet Explorers.

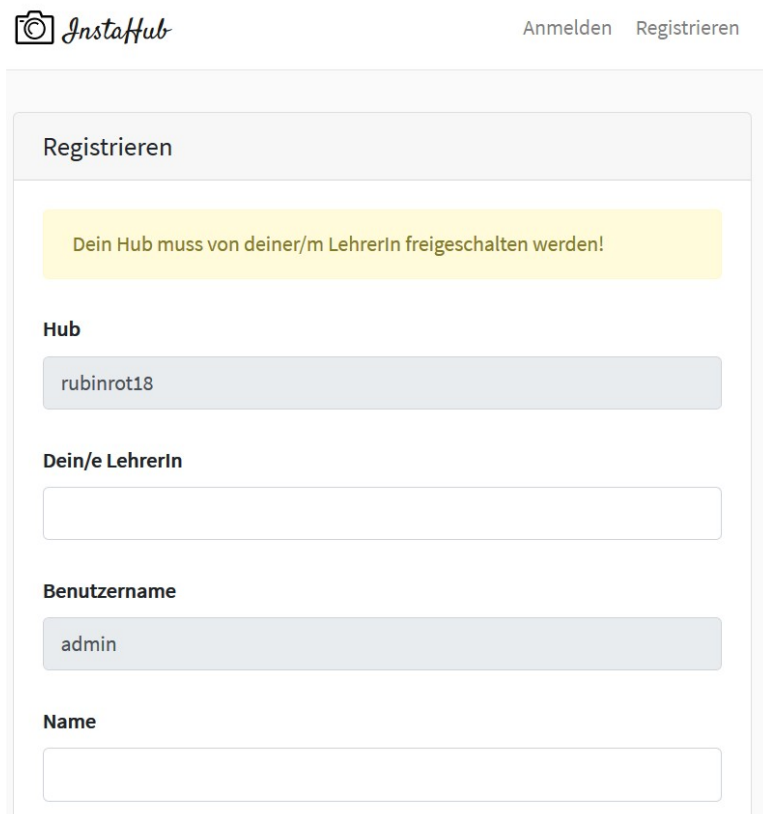
3 Erste Schritte mit InstaHub



The image shows the InstaHub login and registration page. At the top, the 'InstaHub' logo is displayed in a cursive font. Below the logo, a green button labeled 'Hub erstellen' is circled in red. Underneath this button is a horizontal line, followed by the text 'Anmelden'. Below 'Anmelden' are three input fields: 'Hubname', 'Benutzername', and 'Passwort'. Below the 'Passwort' field is a checkbox labeled 'Angemeldet bleiben'. At the bottom of the login section is a blue button labeled 'Anmelden'. Below the blue button is a link labeled 'Passwort vergessen?'.

Einen eigenen InstaHub anlegen - Schritt 2

4. Sie sehen das folgende Formular (Ausschnitt):



The image shows a snippet of the InstaHub registration form. At the top left is the InstaHub logo, and at the top right are the links 'Anmelden' and 'Registrieren'. Below these is a grey header bar with the text 'Registrieren'. Below the header bar is a yellow warning box with the text 'Dein Hub muss von deiner/m LehrerIn freigeschalten werden!'. Below the warning box are four input fields: 'Hub' (containing 'rubinrot18'), 'Dein/e LehrerIn' (empty), 'Benutzername' (containing 'admin'), and 'Name' (empty).

Einen eigenen InstaHub anlegen - Schritt 3 (Ausschnitt)

5. Notieren Sie sich **jetzt** den Namen Ihres InstaHubs (im Beispiel: rubinrot18): _____.

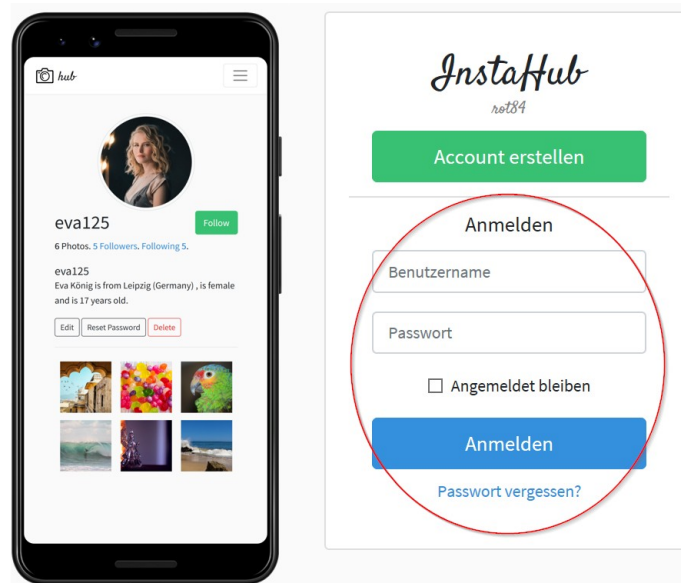
6. *Tragen* Sie unter *Dein/e LehrerIn* den InstaHub-Namen Ihrer Lehrkraft ein. Dies ist _____
7. Unter *Name* tragen Sie bitte Ihre Klasse gefolgt Ihrem mit Ihrer Lehrkraft abgesprochenen Pseudonym ein.
Wenn Ihr Pseudonym Hermine ist und Sie in die Klasse BOM19-3 gehen, tragen Sie ein: BOM19-3 Hermine.
Tragen Sie in den folgenden Feldern Angaben ein wie Sie wollen. Denken Sie sich etwas aus, aber nehmen Sie nicht Ihre echten Daten.
Die E-Mail-Adresse ist allerdings hilfreich, um das Passwort zurücksetzen zu können. Wenn Sie auf diese Funktion verzichten können, brauchen sie Ihre E-Mail-Adresse nicht einzugeben.
8. Wählen Sie ein sicheres Passwort.
9. *Senden* Sie Ihre Registrierung durch einen Klick auf *Register* ab und *warten* Sie nun darauf, dass Ihre Lehrkraft Ihren InstaHub frei schaltet. Erst danach kann es für Sie weiter gehen!

3.2 Am eigenen InstaHub anmelden

Aufgabe 3.2 Verwenden Ihres eigenen InstaHubs

Nachdem Ihre Lehrkraft Ihren InstaHub nun freigeschaltet hat, können Sie Ihren eigenen InstaHub verwenden. Gehen Sie dazu wie folgt vor:

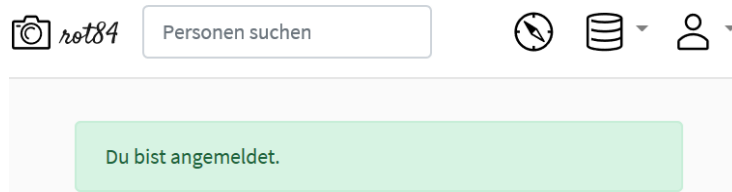
1. Melden Sie sich unter <https://rubinrot18.instahub.org/login> anmelden, wobei Sie *rubinrot18* durch den Namen Ihres InstaHubs ersetzen. Zum Glück haben Sie sich vorhin den Namen Ihres InstaHubs notiert... Ihr Benutzername lautet *admin*.



Login auf Ihrem InstaHub

3 Erste Schritte mit InstaHub

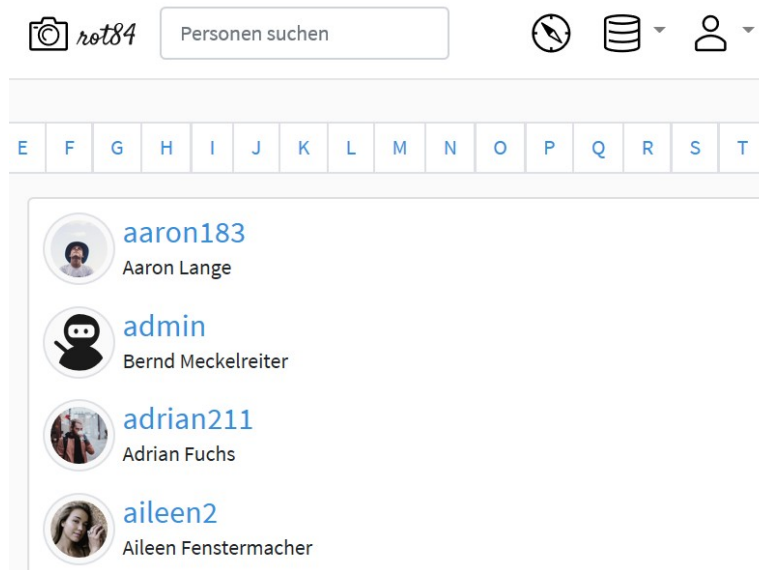
2. Nach dem erfolgreichen Login sehen Sie die derzeit noch recht leere Startseite von InstaHub:



Startseite von InstaHub

Sie erreichen die Startseite immer wieder, wenn Sie auf das **Fotoapparat-Symbol** oben links klicken.

3. Ein Klick auf das **Kompass-Symbol** bringt Sie zur Nutzerübersicht:



Nutzerübersicht

Wie Sie sehen Sie, haben sich ganz ohne Ihr Zutun schon etwa 200 Nutzer*innen an Ihrem InstaHub angemeldet.³

4. Vielleicht werden Sie enttäuscht sein, denn InstaHub ist ja derzeit frei von Fotos. Es gibt irgendwie nur Nutzer*innen, aber die tun nichts. Das kommt noch.

3.3 Nutzer*innen und Administrator*innen

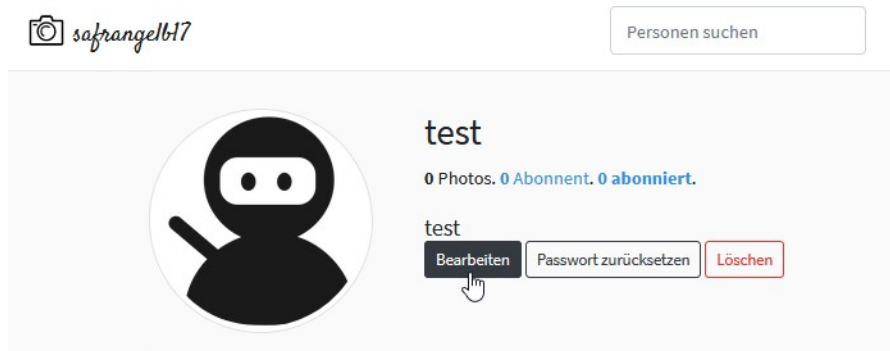
Der Account **admin**, der gemeinsam mit Ihrem InstaHub angelegt wurde, ist ein besonderer Account, der viele Dinge kann, die ein*e normale*r Nutzer*in nicht können darf. Zum Beispiel andere Benutzer*innen löschen. Auch sehen Sie mit Ihrem Administrator-Account Bereiche von InstaHub, die normalen Benutzer*innen verborgen bleiben.

- 3 Keine Angst, diese Nutzer*innen wurden beim Anlegen Ihres InstaHubs angelegt. Es handelt sich **nicht** um echte Menschen. Die Fotos stammen aus Quellen, die den freien Einsatz der Fotos erlauben.

Aufgabe 3.3 Ein eigene*r Nutzer*in

Es wird an einigen Stellen hilfreich sein, wenn Sie in InstaHub auch in die Rolle eine*r normale*n Nutzer*in schlüpfen können. *Erstellen* Sie einen neuen Benutzeraccount für Ihren InstaHub!

1. Melden sie sich als Administrator ab und rufen Sie Ihren InstaHub auf! (z. B. rubinrot18.instahub.org)
2. Wählen Sie Account erstellen und geben Sie wieder minimale Daten ein (s. oben). Wählen Sie ein sicheres Passwort!
3. InstaHub sagt, dass der Account noch freigeschaltet werden muss. Das tun Sie als Administrator: Melden Sie sich nun wieder als Administrator an. Suchen Sie über das Kompass-Symbol und die Suchfunktion Ihren neuen Account.
4. Rufen Sie den Account auf und klicken Sie auf Bearbeiten:



Einen Account bearbeiten

6. Schalten Sie den Account frei, in dem Sie den entsprechenden Haken setzen:

Größe in cm

Avatar

☐ Account ist freigeschaltet

Einen Benutzer freischalten

Aufgabe 3.4 Unterschiede zwischen Benutzer- und Admin-Account

Melden Sie sich abwechselnd mit beiden Accounts an und *vergleichen* Sie deren Arbeitsmöglichkeiten! Wenn möglich, melden Sie sich dazu am besten in zwei verschiedenen Browsern gleichzeitig an. Denken Sie daran, dass der Internet Explorer nicht unterstützt wird.

Aufgabe 3.5: Ausblick

Das Hamburger Abendblatt hat wohlwollend über Ihr soziales Netzwerk be-

richtet. Wie viele der angemeldeten Nutzer*innen wohnen in Hamburg?

Beschreiben Sie, wie Sie vorgehen würden, wenn Sie diese Frage mit Ihrem **bisherigen Kenntnisstand**, quasi “von Hand” beantworten müssten!

3.4 Häufige Fehler beim Anmelden in InstaHub

Achten Sie bitte beim Anmelden immer darauf, dass ...

- Sie auf dem richtigen Hub unterwegs sind, und nicht auf einem falschen oder direkt als Lehrkraft auf instahub.org und
- Sie oben links immer den Namen des verwendeten InstaHubs sehen.
Die Suchmaske

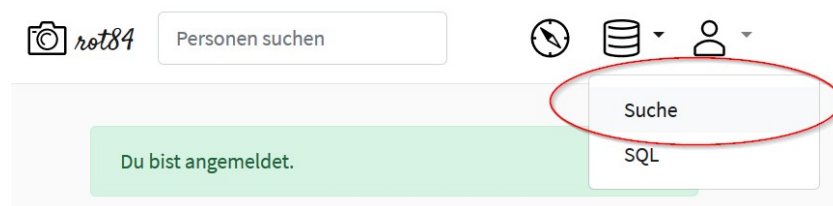
Viele IT-Systeme geben den Benutzer*innen die Möglichkeit, ohne Kenntnis einer Abfragesprache mehr oder weniger einfache Anfragen an das System zu stellen. Sie heißen dort oft “erweiterte Suche”, “Expertensuche” oder ähnliches.

Wir werden die Suchmaske in diesem Skript nur kurz streifen, da wir uns mit der Abfrage- und Manipulationssprache SQL beschäftigen wollen, da diese mächtiger ist.

3.5 Die Suchmaske aufrufen

Sie gelangen wie folgt zur Suchmaske:

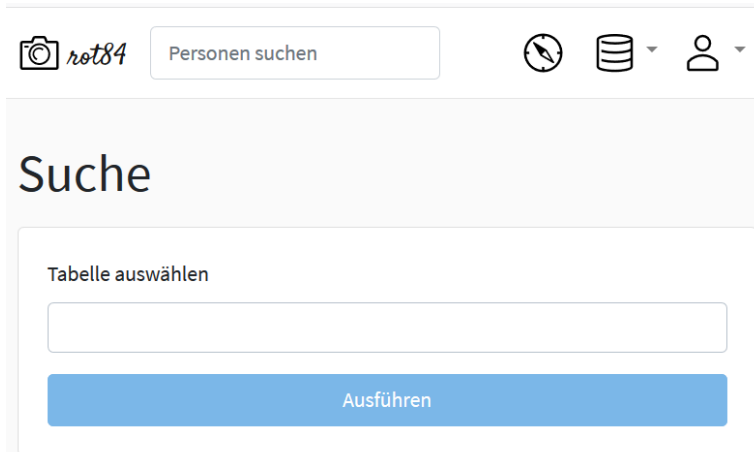
Klicken Sie auf der Startseite oben rechts das Datenbanken-Symbol, hinter dem sich ein Untermenu befindet, aus dem Sie den Punkt Suche auswählen:



Aufruf der Suchmaske

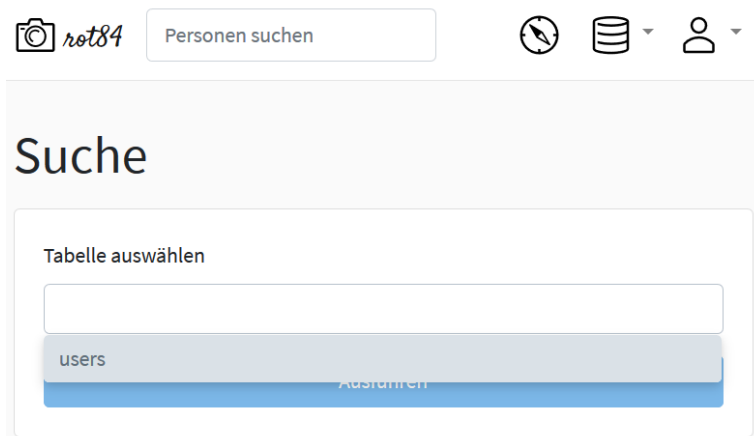
3.6 Die Suchmaske verwenden

Die Suchmaske sieht zunächst sehr aufgeräumt aus:



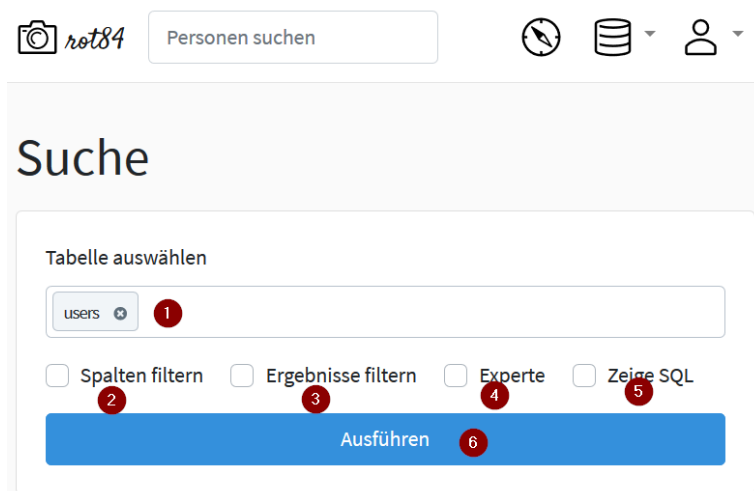
Arbeiten mit der Suchmaske (1)

Wenn Sie in das Feld **Tabelle auswählen** klicken, erscheint eine Drop-Down-Liste, aus der Sie eine Tabelle auswählen können:



Arbeiten mit der Suchmaske (2)

Nachdem Sie eine Tabelle ausgewählt haben, verändert sich die Suchmaske:

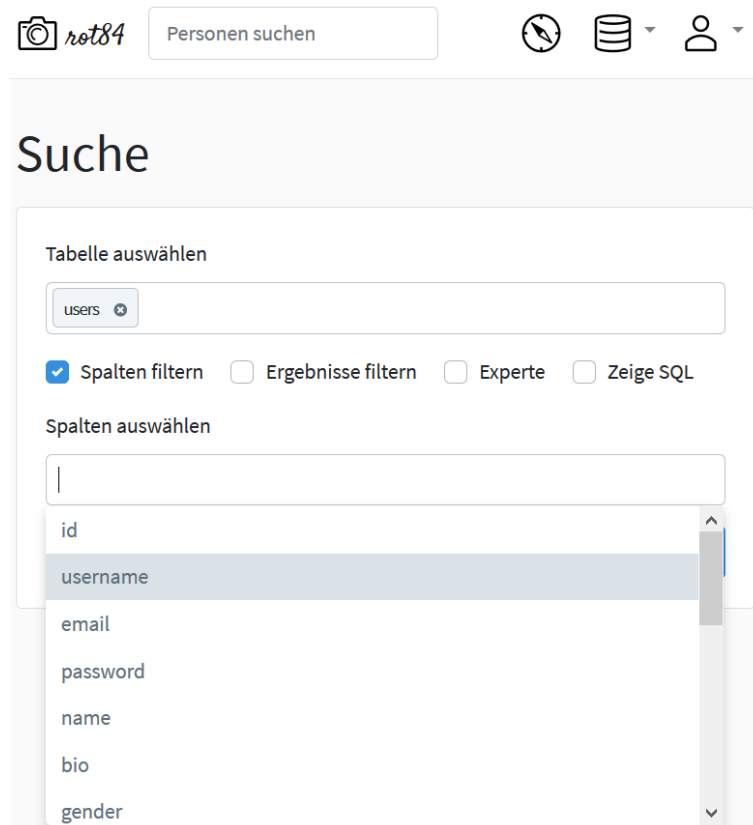


Arbeiten mit der Suchmaske (3)

3 Erste Schritte mit InstaHub

Die einzelnen Elemente in der obigen Abbildung haben folgende Bedeutung:

1. Hier sehen Sie die Tabelle, die Sie ausgewählt haben und auf die sich Ihr Suchauftrag bezieht.
2. Wenn **Spalten filtern** ankreuzen, öffnet sich ein Bereich in dem Sie angeben können, welche der Spalten der Tabelle Sie in Ihre Abfrage einfügen wollen. Diese Auswahl von Spalten wird in Datenbanken auch **Projektion** genannt:



The screenshot shows the InstaHub search interface. At the top, there's a search bar with the text "Personen suchen" and a camera icon. Below the search bar, the title "Suche" is displayed. Under "Suche", there's a section "Tabelle auswählen" (Select table) with a dropdown menu showing "users". Below this, there are four checkboxes: "Spalten filtern" (checked), "Ergebnisse filtern", "Experte", and "Zeige SQL". Below these checkboxes, there's a section "Spalten auswählen" (Select columns) with a dropdown menu showing a list of columns: "id", "username", "email", "password", "name", "bio", and "gender". The "username" column is currently selected and highlighted.

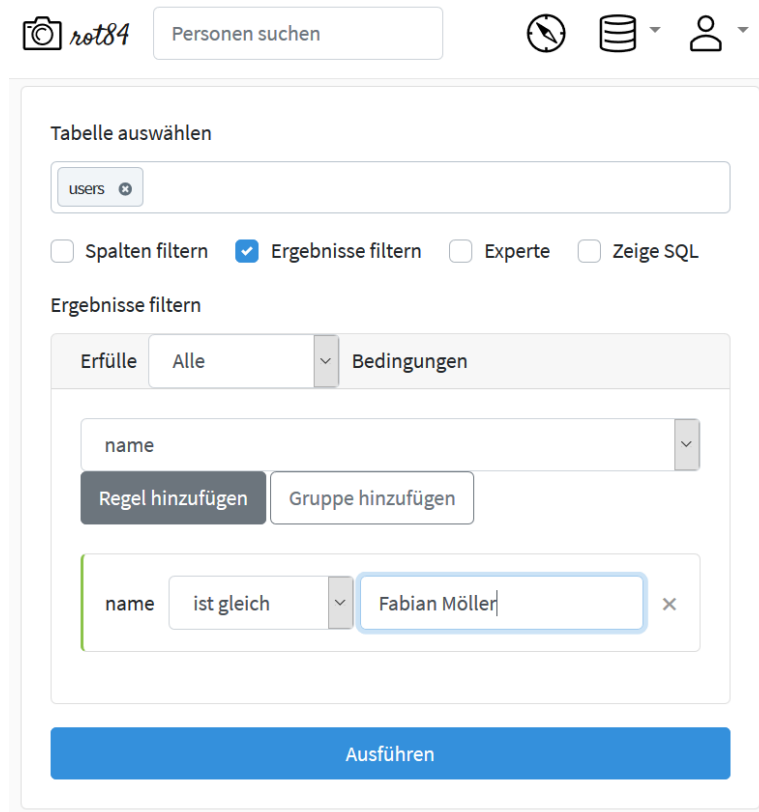
Projektion in der Suchmaske

3. Wenn Sie **Ergebnisse filtern** ankreuzen, können Sie Kriterien angeben, denen alle ausgegebenen Datensätze entsprechen müssen. Man spricht hier von **Selektion**. Die Details entnehmen Sie bitte dem nächsten Abschnitt "[Selektion in der Suchmaske](#)".
4. Das Auswahlfeld **Experte** führt zu einigen Optionen, mit deren Hilfe etwas komplexere Abfragen möglich sind. Diese Möglichkeit wird in diesem Skript nicht behandelt, da wir derartige Abfragen mit der Sprache SQL (siehe Kapitel 4) formulieren werden.
5. **Zeige SQL**: Die in dem Formular eingegebene Suchabfrage wird in die Sprache SQL übersetzt und ausgegeben.
6. **Ausführen**: Die formulierte Suchabfrage wird ausgeführt.

3.7 Selektion in der Suchmaske

Die Selektion, also das Auswählen von Datensätzen aus einer Datenbank an Hand vorgegebener Kriterien ist die vielleicht wichtigste Aufgabe, die ein Datenbankmanagementsystem zu erledigen hat.

Wenn Sie die Daten von Fabian Möller anzeigen wollen, gehen Sie wie folgt vor:



Selektion in der Suchmaske

1. Wählen Sie in der Suchmaske die Tabelle `users` aus.
2. Wählen Sie `Ergebnisse filtern` an.
3. Wählen Sie das Feld `name` aus und geben Sie `ist gleich` aus und tragen sie in das leere Feld den Namen `Fabian Möller` ein.
4. Klicken Sie auf `Ausführen`. InstaHub meldet folgendes Ergebnis:

Anfrage ausgeführt. 1 Ergebnisse gefunden. ×

id	username	email	password
14	fabian141	FabianMoeller@instahub.test	\$2y\$10\$3fm3yriCwXKI

< >

Anzeige des Suchergebnisses in der Suchmaske

Die Suchmaske kann noch mehr: Sie können mehrere Filterregeln angeben. Dann müssen Sie angeben, ob die Suchergebnisse *allen* Filterregeln oder *mindestens einer* entsprechen müssen.

Aufgabe 4.1 Einfache Abfragen mit der Suchmaske

Beantworten Sie die folgenden Fragen mit Hilfe der Suchmaske! *Dokumentieren* Sie Ihre Lösung, z. B. mit Hilfe eines Screenshots.

1. Welche Mitglieder wohnen in Hamburg? Gesucht sind Name, Benutzername und Stadt.
2. Welche Mitglieder wurden nach dem 01.12.1998 geboren? Gesucht sind Name, Benutzername und Geburtstag. (Hinweis: Das Datum müssen sie als 1998 - 12 - 01 eingeben.)
3. Welche Mitglieder sind größer als 190 cm und nach dem 01.12.1998 geboren? Gesucht sind Name, Benutzername und Geburtstag.

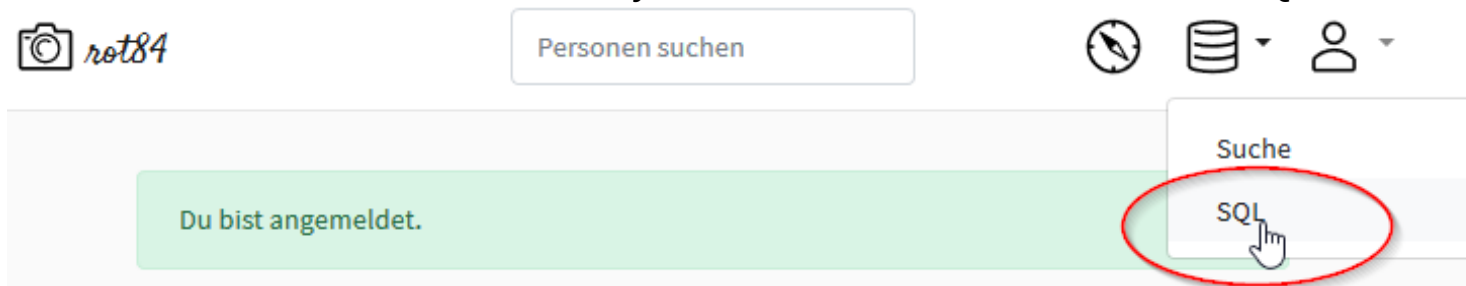
4 Einfache SQL-Abfragen

4.1 SQL kennen lernen

SQL (*structured query language*) ist eine Programmiersprache, mit der man auf Datenbanken zugreifen kann. SQL ist eine Vertreterin der so genannten **deklarativen Programmierung**, da man mit ihr nur das Ergebnis beschreibt, nicht aber den Weg dorthin.

4.1.1 Die erste SQL-Abfrage

1. *Rufen* Sie Ihren InstaHub auf und melden Sie sich als Admin dort an!
2. Klicken oben rechts auf das Symbol *Datenbank* und wählen Sie SQL:



Den SQL-Editor öffnen

3. Sie gelangen in den SQL-Editor:

rot84

Personen suchen

SQL

1

Ausführen

Folgende einzelne Tabellen können abgefragt werden:

password_resets: email, token, created_at

users: id, username, email, password, name, bio, gender, birthday, city, country, centimeters, avatar, role, is_active, remember_token, created_at, updated_at

Der SQL-Editor

- Setzen Sie den Cursor in das leere, im obigen Screenshot rot markierten, Eingabefeld und tippen Sie die folgende SQL-Abfrage exakt so ein:

```
SELECT  username, name, birthday, city
FROM    users
```

Tipps: Nach dem SELECT bzw. FROM tippen Sie die Tab-Taste, damit der Befehl übersichtlich aussieht. Am Ende der ersten Zeile können Sie ENTER drücken, um in eine neue Zeile zu wechseln.

- Klicken Sie auf Ausführen! Sie sehen nun eine Ergebnistabelle mit gut 200 Zeilen:


```
1 SELECT username, name, birthday, city
2 FROM users
```

Ausführen

Anfrage ausgeführt. 204 Ergebnisse gefunden.



username	name	birthday	city
niclas258	Niclas Schweizer	1998-01-31 00:00:00	Wremen
rafael54	Rafael Probst	2001-08-06 00:00:00	Leipzig
luis52	Luis Krüger	2001-12-15 00:00:00	Lautertal
mustav480	Gustav Meister	2001-07-12 00:00:00	Halle

Das Ergebnis Ihres ersten SELECT-Befehls

4.1.2 Tipps:

- Beachten Sie, dass die Spaltennamen exakt mit den Bezeichnungen nach dem SELECT übereinstimmen!
- SQL unterscheidet nicht zwischen Groß- und Kleinschreibung. Es ist aber üblich, die SQL-Befehle in GROSSBUCHSTABEN zu notieren. Halten Sie sich bitte an diese Gepflogenheit.
- Wenn Sie *alle* Spalten ausgeben wollen, können Sie statt alle Spalten aufzuzählen auch einfach ein `*` notieren, z. B.;

```
SELECT *
FROM users
```

- Wenn Sie statt einer Antwort wie der obigen eine englische Fehlermeldung bekommen, haben Sie bei der Eingabe des Befehls einen Fehler gemacht. Kontrollieren Sie noch einmal ganz genau, ob Sie irgendwo einen kleinen Fehler gemacht haben. Sind die Kommata wirklich da, wo sie sein sollen und sind es wirklich Kommata? Kein kleiner Tippfehler in den Spaltennamen?

4.2 SQL-Abfragen verschönern und sortieren

Wenn Sie die Ergebnistabelle ausdrucken wollen, kann es hilfreich sein, die

4 Einfache SQL-Abfragen

Spaltenbezeichnungen verändern zu können. *Probieren* Sie aus, was passiert, wenn Sie statt `birthday` den Text `birthday AS "Geburts-
tag"` eintragen!

```
SELECT  username, name, birthday AS "Geburtstag",  
        city AS "Stadt"  
FROM    users  
ORDER BY birthday ASC  
LIMIT   10
```

Mit der Klausel `ORDER BY` können Sie die Ausgabe sortieren lassen, mit `LIMIT` begrenzen Sie die Zahl der ausgegebenen Datensätze.

4.2.1 Kommentare

Beginnt eine Zeile beginnt mit zwei Minuszeichen (`--`), wird diese vom SQL-Interpreter ignoriert:

```
-- verschönerte Anfrage  
SELECT  username, name, birthday AS "Geburtstag",  
        city AS "Stadt"  
FROM    users
```

Man nennt solche ignorierten Zeilen *Kommentar*. Kommentare sind beim Programmieren wichtig, um zu erklären, was man gerade macht. Zum Beispiel kann man der Lehrkraft mitteilen, welche Aufgabe man bearbeitet. Das ist übrigens auch für Sie selbst hilfreich, wenn Sie (z. B. zur Klausurvorbereitung) noch einmal die alten Aufgaben herausholen.

Es gibt da nur ein Problem: InstaHub verarbeitet nur den ersten Befehl, den man in das SQL-Fenster eingibt. Kommentare sind Befehle. Daher wird der Kommentar "ausgeführt" und nicht die Abfrage.

Also: **Keine Kommentare im SQL-Fenster von InstaHub!**

Wenn Sie mehrere Zeilen als Kommentar markieren wollen, verwenden Sie `/*` und `*/`:

```
/* Verschoenerte Abfrage  
   Liefert alle Nutzer*innen nach Geburtstag sortiert. */  
SELECT  username, name, birthday AS "Geburtstag",  
        city AS "Stadt"  
FROM    users  
ORDER BY birthday ASC  
LIMIT   10
```

4.2.2 ORDER und LIMIT

Statt `ASC` können Sie in der `ORDER`-Klausel auch `DESC` oder gar nichts verwenden. *Beschreiben* Sie, was jeweils passiert!

Beschreiben Sie, welche Bedeutung die LIMIT-Klausel hat!

Aufgabe 5.1: Erste SQL-Abfragen

Erstellen Sie jeweils eine SQL-Abfrage, die die folgenden verbalen Anfragen möglichst schön beantworten:

1. Wer sind die zehn jüngsten InstaHub-User*Innen?
2. Wo wohnen die fünf am längsten angemeldeten InstaHub-User*Innen?
3. Wer ist der erste InstaHub-User*in gewesen?

4.3 Dubletten aussortieren

Uns interessiert brennend, in welchen Städten wir bereits mindestens eine/n Nutzer*In haben. Wir probieren es mit der folgenden Abfrage:

```
SELECT    city
FROM      users
ORDER BY  city ASC
```

Das Ergebnis ist eine lange Liste von Städten, in denen aber einige Städte mehrfach vorkommen.

Probieren Sie es einmal mit

```
SELECT    DISTINCT city
FROM      users
ORDER BY  city ASC
```

Nun werden alle Zeilen aussortiert, die mehr als einmal vorkommen.

Aufgabe 5.2: Dubletten aussortieren

Erstellen Sie jeweils eine SQL-Abfrage, die folgenden verbalen Anfragen möglichst schön beantworten:

1. Welche Körpergrößen haben unser Nutzer*innen?
2. Welche verschiedenen Werte kommen in der Spalte `role` vor?

4.4 Datensätze filtern

Welche InstaHub-Nutzer*Innen leben eigentlich in Dresden?

Wir könnten nun die Liste nach Städten sortieren und dann scrollen, bis wie bei *Dresden* angekommen sind. Das muss einfach gehen. Na klar. Die wohl wichtigste Klausel des SELECT-Befehls fehlt uns noch: `WHERE`

```
SELECT    username, name, city
WHERE     city = "Dresden"
FROM      users
```

Aufgabe 5.3: Deutschlandreise I

Erstellen Sie jeweils eine SQL-Abfrage, die folgenden verbalen Anfragen möglichst schön beantworten:

1. Welche Nutzer*Innen wohnen in Leipzig?
2. Welche Nutzer*Innen haben die Rolle dba?
3. Welche Nutzer*Innen wohnen in Bokholt-Hanredder?
4. Welche Nutzer*Innen wohnen in Hamburg?
5. Ist Justin Schuster bei uns angemeldet?

Aufgabe 5.4 Ups...

Überprüfen Sie Ihre Ergebnisse aus den beiden letzten Abfragen der vorangegangenen Aufgabe noch einmal genau und *begründen* Sie, warum Ihre Ergebnisse offenbar nicht ganz korrekt sind!

4.4.1 Zeichenketten vergleichen

Beim Vergleichen von Zeichenketten ist es oft hilfreich, wenn man nach Teilzeichenketten suchen kann. Hierfür hat SQL den Operator `LIKE`, der anders als `=` nicht die exakte Übereinstimmung erfordert, sondern mit Hilfe von Platzhalten (*Wildcards*) auch Wortteile finden kann. So würde `WHERE name LIKE "Justin%Schuster%"` auch die folgenden Personen finden, wenn Sie denn als Nutzer*innen angemeldet wären:

- Justine Schuster
- Justin Frederik Augustus Frederikus Freiherr von Schuster-Boomzwinkel

Während mit `%` beliebig viele Zeichen (inklusive keinem) gefunden werden, wird mit dem Zeichen `_` (Unterstrich, Shift+Minuszeichen) nur genau ein Zeichen gefunden.

Aufgabe 5.5 Deutschlandreise II

Erstellen Sie jeweils eine SQL-Abfrage, die folgenden verbalen Anfragen möglichst schön beantworten:

1. Welche Nutzer*Innen wohnen in Hamburg?
2. Welche Nutzer*Innen mit dem Namen Schuster sind bei uns angemeldet?
3. Welche Nutzer*Innen mit dem haben wir, die so klingen wie Meier?
(Hier können Sie einen Beifang irrelevanter Datensätze wohl nicht völlig verhindern!)

4.4.2 Numerische Vergleiche

Mit den Vergleichsoperatoren $>$, $>=$, $<$ und $<=$ können auch numerische Werte verglichen werden.

Das funktioniert auch mit Kalenderdaten. `WHERE created_at < "2017-12-31"` liefert alle Benutzer*Innen, deren Datensätze vor dem 31.12.2017 erstellt wurden.

Aufgabe 5.6 Riesen und Zwerge

Erstellen Sie jeweils eine SQL-Abfrage, die folgenden verbalen Anfragen möglichst schön beantworten:

1. Welche InstaHub-Nutzer*Innen sind größer als 188 cm?
2. Welche InstaHub-Nutzer*Innen sind kleiner als 155 cm?
3. Welche InstaHub-Nutzer*Innen sind heute am Tag der Aufgabenbearbeitung *nicht* volljährig?
4. Welche InstaHub-Nutzer*Innen sind älter als 70 Jahre?
5. Gibt es eine/n Nutzer*In, die am selben Tag Geburtstag hat wie Sie?
(Hinweis: Wenn Sie am 12.10.1998 geboren sein sollten, suchen Sie, ob jemand am 12.10.1998 geboren ist!)

5 SQL-Abfragen: Logik und Aggregate

5.1 Logische Operatoren

Aufgabe 6.1 Analyse

Überlegen Sie zunächst, welche Ergebnisse die folgenden SELECT-Befehle liefern könnten! Beschreiben Sie, welche Datensätze in der Ergebnistabelle enthalten sein werden!

```
-- Beispiel 1:
SELECT username, city
FROM users
WHERE city = "Berlin" AND name LIKE "Fabian%"

-- Beispiel 2:
SELECT username, city
FROM users
WHERE city = "Berlin" OR city = "Hamburg"

-- Beispiel 3:
SELECT username, city
FROM users
WHERE city = "Berlin" AND NOT gender LIKE "female"
```

Die folgende Tabelle zeigt die Bedeutung der logischen Operatoren:

A	B	NOT A	A AND B	A OR B	NOT A AND B	NOT(A OR B)
FALSE	FALSE	TRUE	FALSE	FALSE		
FALSE	TRUE	TRUE	FALSE	TRUE		
TRUE	FALSE	FALSE	FALSE	TRUE		
TRUE	TRUE	FALSE	TRUE	TRUE		

Aufgabe 6.2 Logikübung

Vervollständigen Sie die beiden letzten Spalten in der obigen Tabelle!

Aufgabe 6.3 Mehrere logische Operatoren

Erstellen Sie je eine SQL-Abfrage, die

1. alle Berliner auflistet, die *Marc* heißen,
2. alle Leipziger Frauen auflistet,
3. alle Linas und Lorenas auflistet,
4. alle Männer nach ihrer Körpergröße auflistet, die mindestens 16 Jahre alt sind und

5. alle Nutzer liefert, die weder männlich noch weiblich sind.

Aufgabe 6.4 Logische Operatoren mit Klammern

Erstellen Sie je eine SQL-Abfrage (Tipp: Verwenden Sie Klammern wie beim Taschenrechner), die

1. alle Benutzer mit dem Namen "Xaver" auflistet, die nicht in München oder Nürnberg leben und
2. alle Benutzer*Innen auflistet, die 10 cm größer als der Durchschnitt ihrer Geschlechtsgegnossen sind. (Die Durchschnittsgröße von Männern beträgt in Deutschland 179,9 cm, die von Frauen 165,9 cm)

5.2 Zusammenfassungen (Aggregate)

Bisher haben Sie bei Abfragen

- die Spalten, die angezeigt werden sollen, ausgewählt (Projektion) oder
- die Zeilen, die im Ergebnis stehen sollen, ausgewählt (Selektion).

Aber was, wenn wir wissen wollen, wie groß unser größtes Mitglied in jeder Stadt ist? Oder wir für jede Stadt wissen wollen, wie viele Mitglieder wir dort haben?

Die Lösung hierfür heißt Aggregat (aggregieren: lat. *anhäufen*). Die Idee ist in der folgenden Abbildung dargestellt:

username	city
max383	München
jen84	München
malte231	München
karl2	München
karl261	Dresden
brigitte94	Dresden

city	COUNT(city)
München	4
Dresden	2

Aggregate fassen mehrere Datensätze zusammen

Aggregate fassen also mehrere Zeilen der Quelltabellen zu einer einzigen Zeile in der Ergebnistabelle zusammen.

Aufgabe 6.4 Unser erstes Aggregat

Der folgende SELECT-Befehl liefert für jede Stadt den größten und kleinsten Benutzer:

```
SELECT  city AS "Stadt", MIN(centimeters) AS "kleinster",
        MAX(centimeters) AS "groesster"
FROM    users
GROUP BY city
```

Probieren Sie den Befehl aus und *erklären* Sie, wie er funktioniert. *Beachten* Sie dabei *alle* neuen SQL-Wörter.

MAX ist eine **Aggregatsfunktion**. Weitere Aggregatsfunktionen sind beispielsweise COUNT, SUM und AVG (Durchschnitt, engl. *average*).

Aufgabe 6.5 Aggregatsfunktionen

Erstellen Sie je eine SQL-Abfrage, die

1. alle Werte des Feldes gender liefert und angibt, wie oft diese auftreten,
2. die durchschnittliche Größe aller Mitglieder in Dresden liefert und
3. das Geburtsdatum des jüngsten männlichen und des jüngsten weiblichen Mitglieds liefert.

6 CRUD

Aufgabe 7.1 Was ist CRUD?

Recherchieren Sie, was unter CRUD im Zusammenhang mit Datenbanken verstanden wird!

6.1 INSERT - Datensätze einfügen

Überlegen Sie, was der folgende SQL-Befehl vermutlich tun wird!

```
INSERT
INTO users (
  username,          email,          password,
  name,              bio,             gender,
  birthday,          city,            country,
  centimeters,       avatar,          role,
  is_active,          remember_token, created_at,
  updated_at
)
VALUES (
  'guenther37',      'guenther@instahub.app',    '12345',
  'Günther Müller', 'Günther mag Kartoffelsalat.', 'male',
  '2006-06-06 00:00:00', 'Leipzig',    'Deutschland',
  '173',             'avatar.png',              'user',
  '0',               NULL,                       now(),
  now()
)
```

Aufgabe 7.2 Ihr erstes Insert

1. *Geben* Sie den obigen Befehl ein.
2. *Überprüfen* Sie das Ergebnis.
3. *Erklären* Sie, was `now()` bedeutet! (Tipp: Sehen Sie sich an, was in den beiden Spalten in der Tabelle steht!)

Aufgabe 7.3 Noch ein paar Inserts

1. *Fügen* Sie einen Datensatz für Joachim Löw mit allen leicht öffentlich zugänglichen Daten in die Datenbank ein!
2. *Fügen* Sie einen weiteren Datensatz für eine Person Ihrer Wahl in die Datenbank ein!

Aufgabe 7.4 Spezialaufgabe

Überlegen Sie, warum das Feld `id` nicht eingetragen wurde und dennoch gefüllt wurde!

6.2 UPDATE - Datensätze verändern

Um Datensätze zu verändern, gibt es den Befehl `UPDATE`. In einer SQL-Referenz finden Sie folgende Beschreibung:

```
UPDATE < table_name >  
    SET < column1 > = < value1 >, < column2 > = < value2 >, ...  
    WHERE < condition >;
```

Dabei müssen Sie alle in Schreibmaschinenschrift geschriebenen Texte *exakt* so eingeben, wie Sie da stehen. Texte in *< spitzen Klammern >* sind Platzhalter und sollen von Ihnen durch konkrete Inhalte ersetzt werden. Die `WHERE`-Klausel kennen Sie bereits aus dem `SELECT`-Befehl. Und sie funktioniert hier sogar genauso. ;-)

ACHTUNG: Sie können mit einem einzigen `UPDATE`-Befehl viel Unheil anrichten. Wenn Ihnen etwas passiert ist, kontaktieren Sie Ihre Lehrkraft, sie kann Ihre Datenbank zurücksetzen.

Aufgabe 7.5 Update

Erstellen Sie jeweils ein `UPDATE`-Kommando:

1. Alle weiblichen Nutzerinnen sollen 160 cm groß sein.
2. Statt "Germany" soll es im Feld `country` immer "Deutschland" heißen.
3. Ersetzen Sie die Körpergröße bei allen Dresdnern mit dem Wert `FLOOR(RAND() * 45) + 150`.
 1. Ermitteln Sie, was dieser Befehl bewirkt hat!
 2. Recherchieren Sie, was es mit den Funktionen `FLOOR` und `RAND` auf sich hat!
4. Aktivieren Sie den Account von Jogi Löw. (Tipp: Überlegen Sie, welche Spalte für die Aktivierung eines Accounts zuständig sein mag.)
5. Ersetzen Sie die Körpergröße bei allen Münchnern mit einen zufälligen Wert zwischen 130 cm und 190 cm!

6.3 DELETE - Datensätze löschen

Um Datensätze zu löschen, gibt es den Befehl DELETE. In einer SQL-Referenz finden Sie folgende Beschreibung:

```
DELETE FROM < table_name >
WHERE < condition >;
```

Die WHERE-Klausel kennen Sie bereits aus dem SELECT- und dem UPDATE-Befehl.

Achtung: Vor dem Absetzen eines DELETE-Kommandos sollte man immer die verwendete WHERE-Klausel in einem SELECT-Kommando ausprobieren.

Aufgabe 7.6 Delete

Erstellen Sie jeweils ein DELETE-Kommando:

1. Löschen Sie alle nicht aktivierten Accounts.
2. Löschen Sie Jogi Löw.

6.4 Grenzen von CRUD

Kaufleute müssen ihre Geschäfte in Handelsbüchern dokumentieren (Buchhaltung). Heute werden diese Bücher in der Regel nicht auf Papier, sondern in Datenbanken geführt. Im Handelsgesetzbuch wird im §239 Absatz 3 bestimmt:

Eine Eintragung oder eine Aufzeichnung darf nicht in einer Weise verändert werden, daß[!] der ursprüngliche Inhalt nicht mehr feststellbar ist. [...]

Beispiel: Wenn die Bank auf Grund eines Fehlers von Ihrem Konto Geld abbucht und ihren Fehler danach korrigiert, ist die Bank verpflichtet, den Fehler nicht unkenntlich zu machen:

Text	Datum	Betrag	Saldo
DB Hamburg Fahrkarte	22.05.2	19,80-	10,03+
	0		
BarabhebungKiel Hbf	22.05.2	5.052.202,0	5.052.192,0
	0	0-	0-
Storno Wir bitten unser Versehen zu entschuldigen	27.05.2	5.052.202,0	10,03+
	0	0+	

Aufgabe 7.7 Darf der das...?

1. *Begründen* Sie, warum eine vollständige Unterstützung von CRUD in der Buchhaltung nicht zulässig ist.
2. *Finden* Sie Beispiele für Daten, bei denen Sie *vermuten*, dass sie ebenfalls nicht ohne weiteres verändert oder gelöscht werden dürfen.

7 Passwörter speichern

7.1 Was viele Menschen glauben, was beim Login passiert

Wenn Sie sich an einem IT-System anmelden, geben Sie in den meisten Fällen Ihren Benutzernamen und ein Passwort ein. Jede*r, der diese beiden Informationen *weiß*, kann sich als Sie ausgeben (*authentisieren*). Man spricht daher von einem *wissensbasierten System*.

Sie wenden bestimmt Regeln für sichere Passwörter an, unter anderem wissen Sie, dass Sie niemandem Ihr Passwort sagen dürfen. Aber irgendwie muss das IT-System, an dem Sie sich anmelden, ja Ihr Passwort kennen, sonst kann es ja nicht entscheiden, ob das eingegebene Passwort das richtige ist. Man *könnte* also glauben, dass alles ganz einfach ist, und wie folgt abläuft:

1. Die*Der Benutzer*in schickt den Benutzernamen und das Passwort an den Server.
2. Der Server sucht in der Benutzertabelle nach dem Benutzernamen und vergleicht das eingegebene Passwort mit dem gespeicherten.
3. Stimmen beide überein, lässt der Server die*den Benutzer*in rein, sonst nicht.

Diese naive Vorstellung wird auch in der Lach- und Sachgeschichte "Internet" der Sendung mit der Maus präsentiert:



Naiver Passwortvergleich in der Lach- und Sachgeschichte "Internet" der Sendung mit der Maus [<https://www.wdrmaus.de/filme/sachgeschichten/internet.php5>]

Nun haben sie in der Tabelle `users` ja bereits die Spalte `password` gesehen:

username	password
niclas258	\\$2y\\$10\\$VdoILxrn6CZU/ PtClCDJY.eB3s0lzYiGkY7No- kYpZJpMsJ2pq/tuK
rafael54	\\$2y\\$10\\$HkuheWgNzb- w9KLo8N3k0HeM6hh7LTpLwq- M0cAY7QpqKz0i7NIYqUm

Vielleicht haben Sie sich gedacht, dass solche Passwörter sicherlich kein Mensch eingeben würde, aber es sind ja schließlich nur Testdaten. Oder sind das gar nicht die Passwörter? Oder sind sie verschlüsselt?

Aufgabe 8.1 Passwörter im Klartext?

Überprüfen Sie, ob InstaHub die Passwörter im Klartext speichert, indem Sie folgende Tests unternehmen:

1. *Versuchen* Sie, sich an Ihrem InstaHub mit einem beliebigen Benutzernamen und dem zugehörigen Wert in der Spalte `password` anzumelden.
2. *Überprüfen* Sie, ob das von Ihnen verwendete `admin`-Passwort in der Datenbank steht.
3. *Überprüfen* Sie, ob sich der Wert des `admin`-Passworts ändert, wenn Sie das Passwort über das Profil ändern.

Aufgabe 8.2 Admins Probleme mit Passwörtern

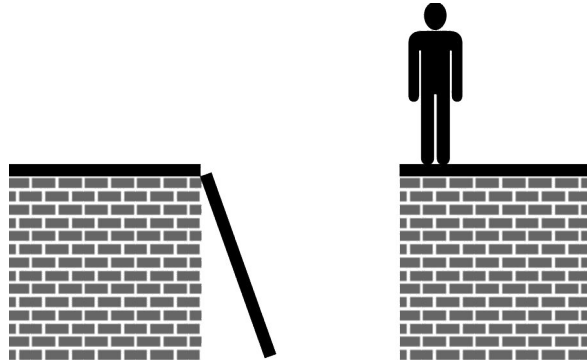
1. *Recherchieren* Sie, wann in der letzten Zeit einem Server unverschlüsselte Passwörter abhanden gekommen sind!
2. *Finden* Sie Gründe, die dafür sprechen, Passwörter nicht unverschlüsselt zu speichern.
3. *Finden* Sie Gründe für die Forderung, dass Passwörter so gespeichert werden müssen, dass es auch den Administratoren nicht möglich ist, deren Klartext wieder herzustellen!

7.2 Was wirklich beim Login passiert

Das klingt merkwürdig. Man muss das Passwort vergleichen können, aber zugleich soll man es nicht speichern. Wie soll das nur gehen? Zeit für einen kleinen Exkurs:

7.2.1 Falltüren

Es ist sehr einfach durch die Falltür nach unten zu gelangen, während sich der Weg zurück manchmal als ausgesprochen schwierig, wenn nicht gar unmöglich, erweist:



Durch eine Falltür geht es nur in eine Richtung leicht hindurch...

Diese Idee kann man auf das Speichern von Passwörtern übertragen. Wir schicken das Passwort durch die Falltür und speichern das Ergebnis. Bei der Anmeldung schicken wir das eingegebene Passwort ebenfalls durch die Falltür und vergleichen dann das Ergebnis mit dem gespeicherten Wert. Stimmen beide überein, war das Passwort richtig. Klingt erstmal merkwürdig.

Wie Sie wissen, können Computer nur Zahlen speichern. Auch Texte werden intern als Zahlen gespeichert. Und mit denen kann man rechnen. Mathematiker*innen kennen so genannte *Einwegfunktionen*. Die Funktionswerte solcher Einwegfunktionen sind zwar verhältnismäßig leicht auszurechnen, aber es ist fast unmöglich in der Gegenrichtung aus einem Funktionswert zurück zum eingegebenen Wert zu kommen.

Als Beispiel für eine Funktion die einer Einwegfunktion ähnelt, kann die Primfaktorzerlegung dienen. Die Zerlegung der Zahl 21 in ihre Primfaktoren ist leicht: 7 und 3. Die Primfaktorzerlegung wird bei sehr großen Zahlen aber sehr zeitaufwendig. Versuchen Sie doch mal, die Zahl 162.249 in ihre Primfaktoren zu zerlegen! Sie werden ziemlich lange suchen müssen, bis Sie eine Lösung finden. Ihnen bleibt nur der Reihe nach alle Primzahlen ausprobieren. Umgekehrt ist es für Sie aber ein Leichtes, die Aufgabe $433 \cdot 443$ zu lösen. Das Ergebnis ist: 162.249. Wenn man jetzt die Zahlen noch viel, viel größer wählt, wird die Primfaktorzerlegung ein sehr großes Problem.

In der Informatik nennt man diese Funktionen **Hashfunktionen** (engl. *hash*: **Hackfleisch**). Falls Sie sich fragen, warum so etwas Hackfleischfunktion heißt: Spielen Sie die Falltür-Idee mit den Begriffen Katze und Fleischwolf durch ...

Nennen wir die Hashfunktion $h(p)$. Dann läuft unsere Anmeldeprozess so ab:

1. Der*die Benutzer*in gibt ihren*seinen Benutzernamen und das Passwort p ein.
2. Der Server bildet $h(p)$ und vergleicht das Ergebnis mit dem gespeicherten Wert.
3. Sind die beiden Werte gleich, war die Authentisierung erfolgreich.

Aufgabe 8.3 Der Wert der Hashtabelle

Durch Umstände, die wir an dieser Stelle nicht weiter ergründen wollen, sind suspekte Individuen aus Ihrem Informatik-Kurs in den Besitz der Passwort-tabelle des Schulverwaltungsprogrammes gekommen. *Erklären* Sie, warum es ihnen nicht ohne weiteres möglich sein wird, sich am Schulverwaltungsprogramm als Lehrkraft auszugeben!

7.2.2 Gebräuchliche Hashfunktionen

Gebräuchliche Hashfunktionen sind u. a. MD5, SHA-1, SHA-256 oder bcrypt. Letztere wird übrigens von InstaHub verwendet. Die Hashfunktionen bilden aus jeder übergebenen Zeichenkette einen Hashwert fester Zeichenlänge. Egal ob Sie “Hallo Welt!” oder die ganze Bibel hashen - es kommt eine wüste Zeichenkette stets derselben Länge heraus, schon eine geringfügige Änderung der Eingabe führt zu einem völlig anderen Hash-Wert:

Text	Hash-Wert (SHA-1)
Die Bibel ⁴	0f1a3d197eb98fb4638e - f02866159160269b89b8
“Hallo Welt!”	726c3e8861ab0652a5043ea5f aff6d3ef33fb209
“Hallo Welt?”	dff4b83f415b61a06594d - c57c4972c77da0c833f

Es gibt also mit Sicherheit mehrere Texte, die zu demselben Hashwert wie “Hallo Welt!” führen. Eine sichere Hashfunktion erkennt man daran, dass man diese so genannten *Kollisionen* nur zufällig finden kann.

Aufgabe 8.4 Kollisionen

Nehmen wir an, als (völlig ungeeignete) Hashfunktion wird verwendet “Nimm die ersten zwei und die letzten zwei Zeichen.”

1. *Ermitteln* Sie den Hashwert des Passwortes Geheim!.
2. *Finden* Sie ein kollidierendes Passwort zu Geheim!.
3. *Erklären* Sie, warum es problematisch ist, wenn sich Kollisionen anders als durch Ausprobieren finden lassen!

7.2.3 Angriffsmöglichkeiten auf Ihr Hackfleisch

Auf den ersten Blick haben wir alles getan, um die Passwörter unserer Benutzer*innen zu schützen. Aber...

Bei aller Sorgfalt kann es überall passieren, dass die Tabelle mit den gehashten Passwörtern abhanden kommt. Wir müssen also damit rechnen, dass sich jemand in aller Ruhe zu Hause hinsetzen kann und die Tabelle unter die Lupe nimmt.

Die*der Angreifer*in kann nun sehr viele verschiedene Passwörter ausprobieren und $h(p)$ bilden. Wenn der Hashwert in der Tabelle enthalten ist, hat sie*er ein Passwort geknackt. Dieser Angriff kann sowohl in Form eines *Wörterbuchangriffs* als auch in Form eines *Brute-Force-Angriffs* erfolgen.

Aufgabe 8.5 Angriffsarten

Erklären Sie, was ein Wörterbuchangriff und was ein Brute-Force-Angriff ist.

7.2.4 Salz und Pfeffer erschweren Angriffe

Der Einsatz einer Hashfunktion alleine reicht also leider nicht aus. Wir müssen das entstandene Hackfleisch noch ordentlich würzen. Hierzu verwenden wir Salz (*salt*) und Pfeffer (*pepper*). Informatiker*innen mögen offenbar keine Zwiebeln.



*Informatiker*Innen mögen offenbar Hackfleisch, aber nur ohne Zwiebel! (Bild von Andreas Lischka auf Pixabay)*

Der oben beschriebene Angriff auf die Hashtabelle war ja erfolgversprechend, weil man jeden erzeugten Hashwert in vielen Zeilen und vielen Tabellen suchen konnte. Wenn wir dieses Problem beheben können, sind wir sehr viel weiter. Man kann sich das Vorgehen wie folgt vorstellen:

1. An das Passwort wird ein von dem*von der Benutzer*in abhängiger Text (das *salt*) angehängt, z. B. der Benutzername. Wenn man das Passwort jetzt hasht und speichert, muss die*der Angreifer*in also den Hashwert für jede*n Benutzer*in einzeln ermitteln. Das kostet viel Zeit.
2. An das um das *salt* verlängerte Passwort hängt der Server vor dem hashen aber noch einen Text an, ein geheimes *pepper*. Er bildet also $h(\text{password} + \text{salt} + \text{pepper})$. Jetzt kann der*die Angreifer*in nicht einmal mehr für mehrere Server zugleich die Hashwerte ermitteln, sondern muss dies für jeden Server einzeln tun. Und dafür braucht er*sie auch noch Kenntnisse über *pepper* und *salt* und vor allem: sehr viel Zeit!

Aufgabe 8.6 Salz und Pfeffer

Ein Server verwendet zur Speicherung der Passwörter SHA-256. Als *salt* wird der Benutzername verwendet. Das *pepper* ist Kvi1lbuche. Der Hashwert wird in der Form $h(p \& \text{salt} \& \text{pepper})$ ermittelt. Wobei $\&$ für die Aneinanderreihung von Zeichenketten steht. 2. Der*Die Nutzer*in nic-las258 meldet sich mit dem Passwort TaKaLu908#an. *Ermitteln* Sie (z. B. auf www.hashgenerator.de) den zugehörigen Hashwert! 3. *Überprüfen* Sie Ihre Lösung, indem Sie den folgenden QR-Code entschlüsseln:



Lösung Aufgabe 8.6 als QR-Code

8 Tabellen erstellen und das Entity-Relationship-Modell

8.1 CREATE TABLE

Aufgabe 9.1 Das erste CREATE TABLE verstehen

Erklären Sie was der folgende SQL-Befehl genau tut. Ziehen Sie dabei das Glossar im folgenden Abschnitt [Glossar zu SQL-Create](#) (oder eine SQL-Referenz Ihres Vertrauens) zu Rate.

Tipp: Es ist vielleicht zunächst hilfreich herauszufinden, wie die neue Tabelle und ihre Spalten heißen werden.

```
CREATE TABLE photos (
  id          INT(4) UNSIGNED NOT NULL      AUTO_INCREMENT,
  user_id     INT(4) UNSIGNED NOT NULL,
  description VARCHAR(255)    NOT NULL,
  url         VARCHAR(255)    NOT NULL,
  created_at  TIMESTAMP      NOT NULL      DEFAULT now(),
  updated_at  TIMESTAMP      NOT NULL      DEFAULT now(),
  PRIMARY KEY (id),
  FOREIGN KEY (user_id)
    REFERENCES users(id)
    ON DELETE CASCADE
)
```


8.1.1 Glossar zu SQL-Create

- **AUTO_INCREMENT** bedeutet, dass bei jedem neu eingefügten Datensatz dieser Wert automatisch um 1 hochgezählt wird. Darum braucht man sich also nicht zu kümmern.
- **DEFAULT** regelt, welchen Wert ein Datensatz bekommt, wenn beim Anlegen des Datensatzes kein Wert angegeben wird. (engl. *default*: Standard, Vorgabe)
- **FOREIGN KEY**: Diese Spalte ist ein Verweis auf eine andere Tabelle. Jeder Wert in dieser Spalte befindet sich in einer Spalte der Zieltabelle. Diese Spalte heißt in der aktuellen Tabelle **Fremdschlüssel**. Die Tabelle, auf die verwiesen wird, heißt **Zieltabelle**. Bei InstaHub ist die Zieltabelle die Tabelle `users` und die Spalte, in der dort gesucht werden soll ist die Spalte `id`. Wie fast immer ist der referenzierte Fremdschlüssel in der Zieltabelle der dortige Primärschlüssel.

lüssel (s. PRIMARY KEY).

Jeder Wert eines Fremdschlüssels muss zwingend in der Zieltabelle enthalten sein. Das kontrolliert die Datenbank. Und damit sie das kann, deklariert man Fremdschlüssel als FOREIGN KEY. In der Tabelle Städte verweisen die Werte in der Spalte Land_ID auf die die Tabelle Länder und zwar auf die Spalte Land_ID. In der Tabelle Länder ist Land_ID der **Primärschlüssel**. In der Tabelle Städte ist die Spalte Land_ID ein **Fremdschlüssel**, der auf eine Zeile in einer anderen Tabelle verweist.

Das Prinzip sehen Sie im folgenden Schaubild:



Stadt_ID	Name	Land_ID	Einwohner
1	Elmshorn	1	49618
2	New York	2	853673
3	Leipzig	1	581980

Land_ID	Name	Einwohner
1	Deutschland	83000000
2	USA	327774344

Fremdschlüssel verweisen auf eine andere Tabelle

- INT ist ein Datentyp. Jede Spalte kann nur Werte einer bestimmten Art enthalten. Es geht also nicht, dass in einer Spalte mal eine Zahl, mal ein Text und dann wieder ein Datum steht. INT (4) bedeutet, dass in der Spalte ganze Zahlen (engl. *integer*) von -2.147.483.648 bis +2.147.483.647 stehen können. Da sie hier aber als UNSIGNED deklariert wurden, sind sie immer positiv und können von 0 bis 4.294.967.295 reichen. Die Zahl in Klammern ist von untergeordneter Bedeutung. Sie gibt an, wie viele Stellen zur Ausgabe der Zahl bereit gestellt werden sollen.
- NULL bedeutet nicht "0", sondern "nichts, leer". NULL ist nicht gleich etwas anderem, auch nicht dem Wert 0. Wenn eine Tabellenspalte in jeder Zeile einen Wert haben soll, kann man NOT NULL notieren, dann wird die Datenbank dort unter keinen Umständen erlauben, nichts einzutragen. Notiert man NULL, erlaubt man auch leere Tabellenzellen.
- ON DELETE: Dies ist Teil der Deklaration eines Fremdschlüssels, siehe FOREIGN KEY. Diese Regel beantwortet die Frage, was passiert, wenn der Zieldatensatz gelöscht wird. ON DELETE CASCADE bedeu-

tet, dass beim Löschen eines Datensatzes in der Tabelle `USERS` alle zugehörigen Datensätze in der Tabelle `photos` gelöscht werden. Würde man dies nicht vereinbaren, müssten zunächst alle Fotos des*er Benutzer*in gelöscht werden, damit der Datensatz in der Tabelle `users` gelöscht werden kann.

- **PRIMARY KEY:** Vereinfacht gesehen hat jede Tabelle eine Spalte, in der kein Wert doppelt vorkommt. Diese Spalte nennt man **Primärschlüssel**. Wenn man hier nach einem Wert sucht findet man also immer keine oder genau eine Zeile, niemals aber mehrere. Mit der Deklaration **PRIMARY KEY** geben Sie an, welche Spalte der Primärschlüssel ist.
- **TIMESTAMP** ist ein Datum mit Uhrzeit. Das frühest mögliche Datum ist 1970, das spätest mögliche 2038.
- **VARCHAR** ist ebenfalls ein Datentyp (s. **INT**). Bei **VARCHAR** handelt es sich um eine Zeichenkette variabler Länge. Die maximale Länge wird in Klammern angegeben.

8.1.2 Weitere SQL-Datentypen

SQL verfügt über zahlreiche Datentypen. Leider wird nicht jeder Datentyp von jedem SQL-Server unterstützt oder gleich interpretiert. Die Datentypen in der folgenden Auswahl wird zwar in diesem Kapitel nicht verwendet, ist aber trotzdem wichtig:

Datentyp	Bedeutung
<code>DECIMAL(p, s)</code>	Eine Zahl mit festem Nachkommateil. <i>s</i> gibt die Zahl der Nachkommastellen an, <i>p</i> die Zahl der Stellen insgesamt. Der Datentyp ist sehr geeignet für Geldbeträge.
<code>CHARACTER LARGE OBJECT</code>	Wie VARCHAR , aber es gibt keinerlei obere Grenze, so dass ganze Textdokumente gespeichert werden können. Umständliche Handhabung, daher wird VARCHAR bevorzugt.
<code>FLOAT</code>	Zahl mit gleitend vielen (engl. <i>to float</i>) Nachkommastellen, wie beim Taschenrechner. Achtung: Es kommen Rundungsungenauigkeiten vor. Der Datentyp ist geeignet für technische Zahlen und nicht für Geldbeträge.
<code>SMALL INT</code>	Kleiner Ganzzahlwert (meist -32.768 bis +32.767)
<code>CHAR(n)</code>	Zeichenkette der festen Länge <i>n</i> . Die Nutzung ist weniger aufwändig als VARCHAR .

Aufgabe 9.2 CREATE TABLE photos

1. *Erzeugen* Sie die Tabelle `photos` so wie zu Anfang dieses Kapitls angegeben. Achten Sie besonders auf die Datentypen und die Namen der Spalten.
2. *Füllen* Sie die Tabelle mit Hilfe der von Ihrer Lehrkraft bereitgestellten Datei mit Daten!
3. *Schauen* Sie sich in Ihrem InstaHub um!
4. *Überlegen* Sie sich, wie die Fotos in den Hub gekommen sind. Sie haben ja schließlich nur eine Textdatei hochgeladen...

8.2 Das Entity-Relationship-Modell

Wir haben nun drei Tabellen in unserer Datenbank. Um den Überblick zu behalten, wäre es schön, die Tabellenstrukturen grafisch darzustellen. Hierzu gibt es das Entity-Relationship-Modell, kurz **ER-Modell** oder ERM. Ein ER-Modell besteht aus den folgenden vier Elementen:

1. **Gegenstände (Entities)**: Eine *Entität* (engl. *entity*) stellt eine Klasse von *Objekten* der *realen* Welt im ER-Modell ab. Z. B. Produkt, Kunde, Schüler oder Informatikkurs. (Sie sehen, dass Objekte nicht unbedingt gegenständlich sein müssen.) Entities werden in ER-Modellen als Rechtecke dargestellt:



Schüler

Informatikkurs

Entitäten werden im ERM als Rechtecke dargestellt

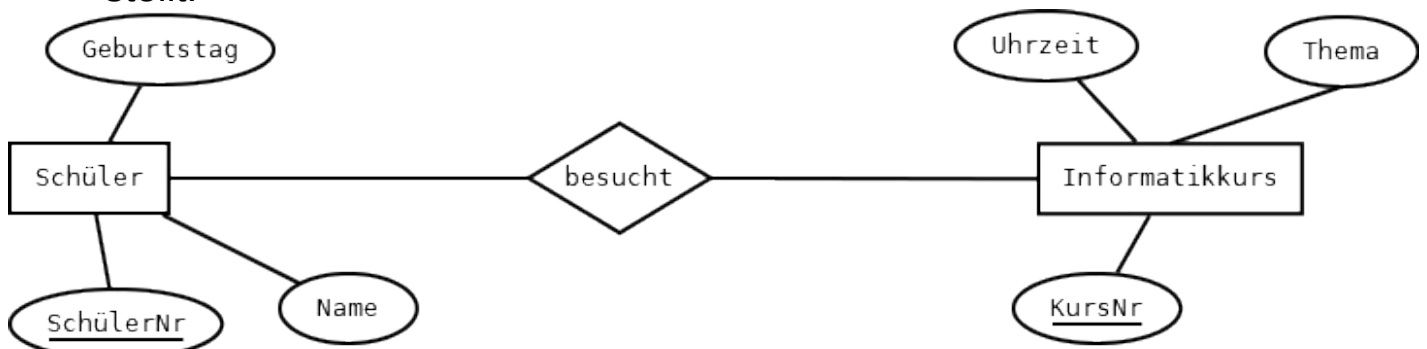
2. **Attribute**: Entitäten und Beziehungen haben *Eigenschaften*, die durch *Attribute* beschrieben werden. Diese Attribute findet man in der Datenbank in den Tabellenspalten. Typische Eigenschaften für die Entität Produkt könnten z. B. Bezeichnung, Gewicht, Preis und Artikelnummer sein.

Attribute werden in ER-Modellen als Ovale an die Entitäten angehängt. Die Primärschlüssel heißen im ER-Modell eigentlich *identifizierende Attribute* und werden unterstrichen:



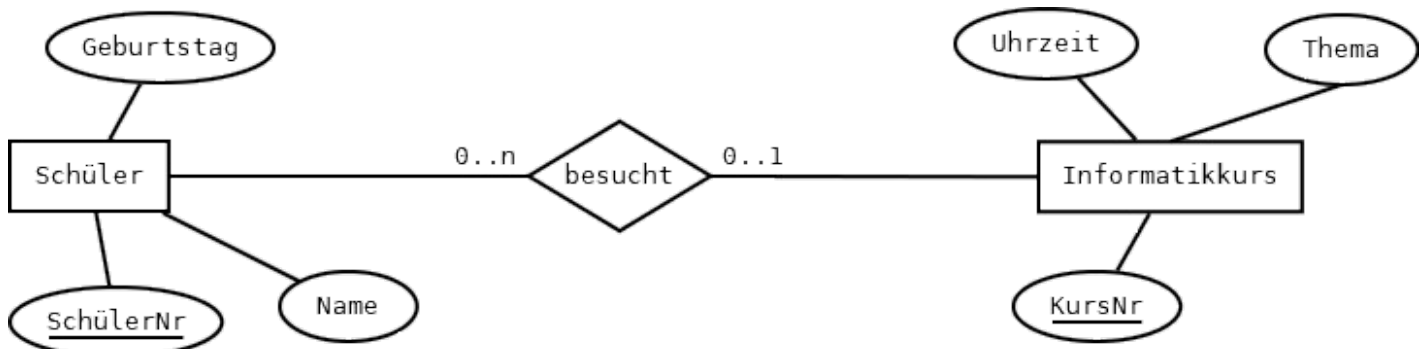
Attribute werden als Ovale angehängt

3. **Beziehungen (Relationships):** Ein *Relationship* stellt eine Beziehung zwischen den Objekten der realen Welt dar. Z. B. die Beziehungen *kaufte* zwischen Kunde und Produkt *besucht* zwischen den Entitäten Schüler und Informatikkurs. Relationships verbinden *immer* genau zwei Entitäten. Relationships werden als Raute dargestellt:



Relationships werden durch Rauten dargestellt

4. **Kardinalität:** Beziehungen verbinden nicht immer genau ein Exemplar einer Entität mit genau einem Exemplar einer anderen Entität. So kann ein Kunde mehrere Produkte gekauft haben. Kardinalitäten werden leider sehr verschieden notiert. Wir verwenden hier die in der folgenden Abbildung verwendete Notation: Ein bestimmte*r Schüler*In hat genau einen (1 . . 1) Informatikkurs und jeder Informatikkurs wird von 0 bis beliebig vielen Schüler*Innen (0 . . n) besucht.)



Kardinalitäten geben an, wie viele Exemplare an dem Relationship beteiligt sind

Bitte beachten Sie: Die **Fremdschlüssel** werden im ER-Modell *nicht* notiert, nur die **Primärschlüssel**. Die Fremdschlüssel werden erst beim Umsetzen der ER-Modells in Tabellen ergänzt. Im obigen Beispiel müsste in der Tabelle Informatikkurs der Primärschlüssel der Entität Schüler, SchülerNr eingefügt werden. Wir werden uns in Abschnitt 10.3 und in Kapitel 17 näher mit diesem Thema beschäftigen.

Aufgabe 9.3 Das ER-Modell von InstaHub (1)

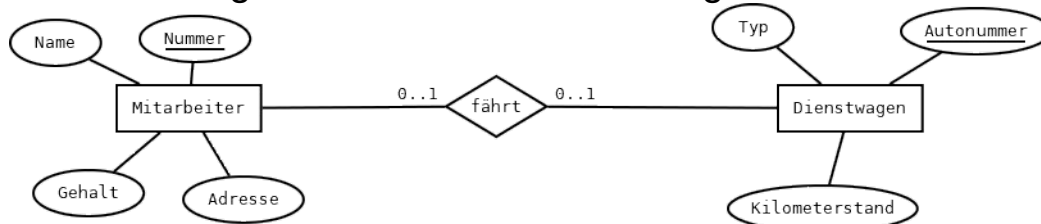
Erstellen Sie ein ER-Modell Ihrer InstaHub-Datenbank! Bei den Attributen wählen Sie einige exemplarische aus.

9 Vertiefung Entity-Relationship-Modell

9.1 ER-Modelle besser verstehen

Aufgabe 10.1 ER-Diagramm beschreiben - Dienstwagen

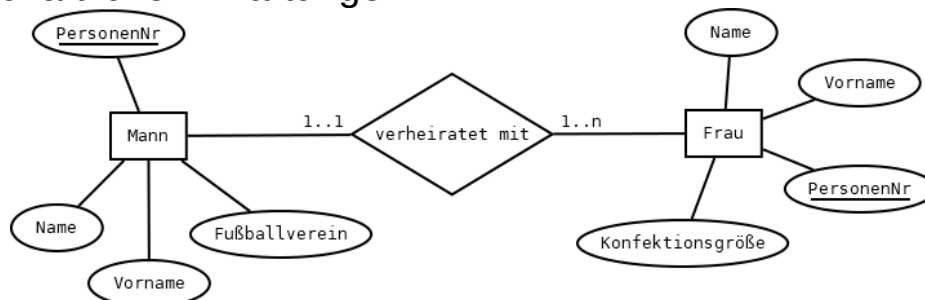
Beschreiben Sie in eigenen Worten, was das folgende ER-Modell modelliert!



*Mitarbeiter*innen fahren Dienstwagen*

Aufgabe 10.2 ER-Diagramm interpretieren Mann und Frau

Beschreiben Sie in eigenen Worten, was das folgende ER-Modell modelliert. Beurteilen Sie es! Berücksichtigen Sie dabei die deutsche Rechtsordnung und gesellschaftliche Erwartungen!



Männer sind mit Frauen verheiratet

9.2 Kardinalitäten verwenden

Aufgabe 10.3 Kardinalitäten festlegen

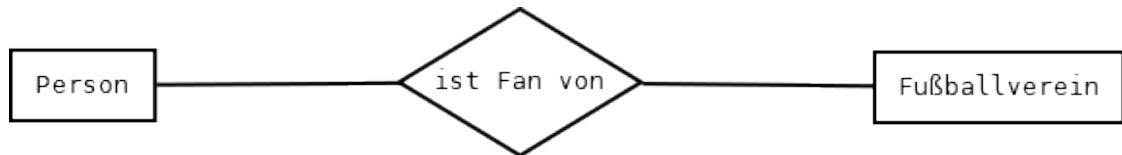
Notieren Sie in den folgenden ER-Diagrammen jeweils die Kardinalitäten!



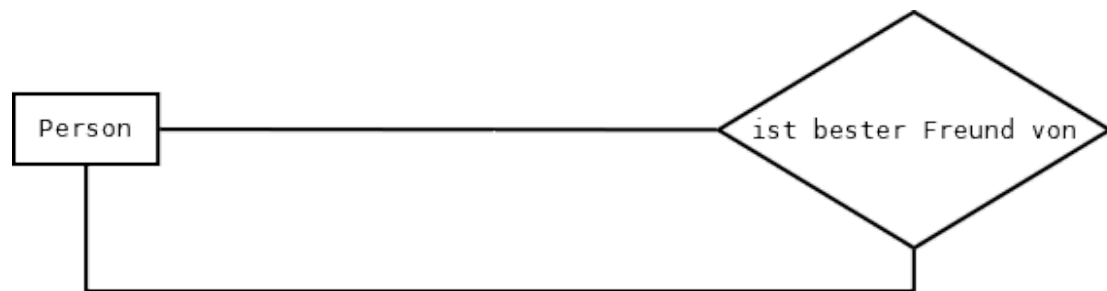
Männer sind Väter von Kindern



Männer sind Geliebte von Frauen



Personen sind Fans von Fußballvereinen



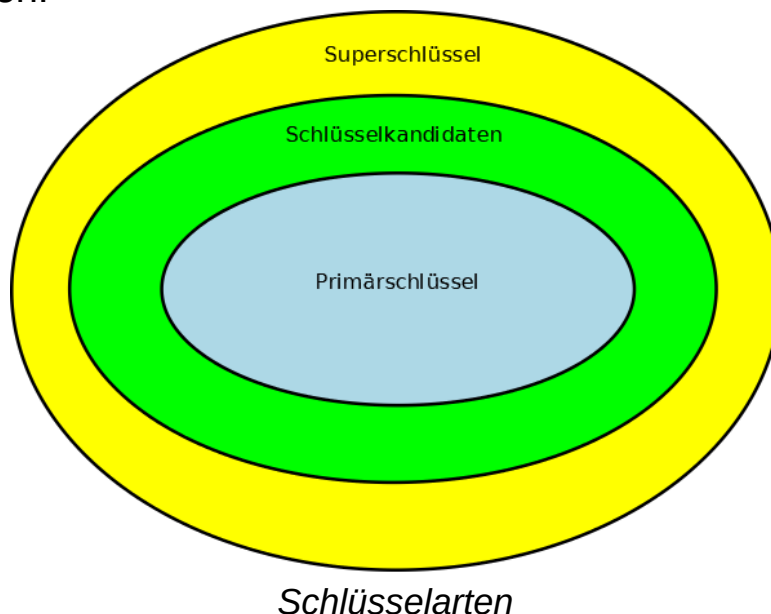
Personen sind bester Freunde von Personen

9.3 Schlüssel

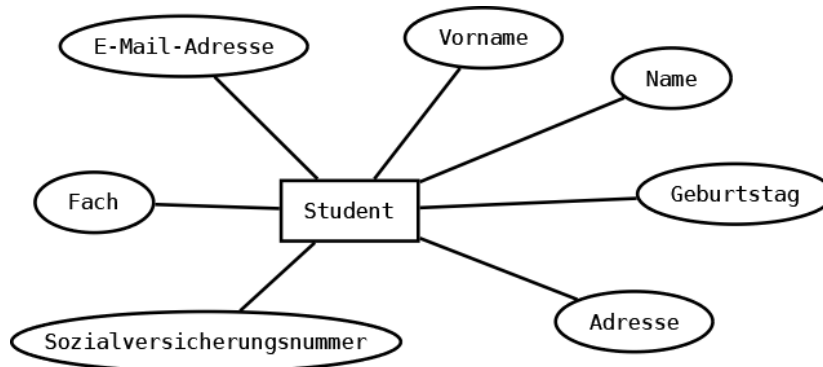
Alle Exemplare einer Entität müssen sich hinsichtlich ihrer Attribute unterscheiden. Es darf also keine zwei identischen Exemplare geben.

Schlüssel sind Mengen von Attributen, die ein Exemplar einer Entität identifizieren, als eindeutig bestimmen.

Dabei werden die in der folgenden Abbildung ersichtlichen Schlüsselarten unterschieden:



Diese Schlüsselarten sehen wir uns an Hand eines Beispiels genauer an. Die folgende Abbildung zeigt einen Ausschnitt eines ERM für die Verwaltung einer Hochschule:

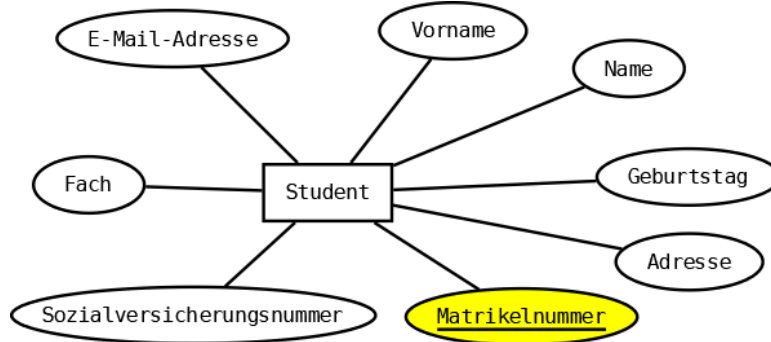


ER-Modell einer Hochschulverwaltung (Ausschnitt)

- **Superschlüssel** (auch "Oberschlüssel") sind *alle* Attributmengen, die identifizierend sind, den*die Studierende*n also eindeutig bestimmen. Es gibt also fast immer mehrere Superschlüssel:
 - Da jedes Exemplar der Entität anders sein muss, ist die Menge aller Attribute schon mal ein (trivialer) Superschlüssel: {E-Mail, Vorname, Name, Geburtstag, Adresse, Sozialversicherungsnummer, Fach}
 - Streicht man das studierte Fach aus dem Superschlüssel, bleibt der Rest der Attribute identifizierend und damit Superschlüssel: {E-Mail, Vorname, Name, Geburtstag, Adresse, Sozialversicherungsnummer}
 - Auch die Adresse kann man aus dem Superschlüssel entfernen: {E-Mail, Vorname, Name, Geburtstag, Sozialversicherungsnummer}
- Ein Superschlüssel, aus dem man kein Attribut mehr entfernen kann, ohne dass er kein Superschlüssel mehr wäre, heißt **Schlüsselkandidat**. In unserem Beispiel sind das:
 - {Vorname, Name, Geburtstag, Adresse}, wenn wir davon ausgehen, dass es bei großen Gebäuden unter derselben Adresse durchaus zwei Kevin Schulzes geben könne, die am selben Tag geboren wurden. Ein solcher Schlüssel, der aus mehreren Attributen zusammengesetzt ist, heißt **zusammengesetzter** Schlüssel.
 - {E-Mail-Adresse}, wenn wir davon ausgehen, dass jede*r Studierende eine eigene E-Mail-Adresse verwendet.
 - {Sozialversicherungsnummer}, wenn wir davon ausgehen, dass jede*r Studierende eine Sozialversicherungsnummer hat.
- Aus der Menge der Schlüsselkandidaten sucht sicher der*die Datenbankdesigner*in einen aus, den er*sie zum **Primärschlüssel** ernennt. Da man den Primärschlüssel auch mal eingeben muss, sind zusammengesetzte und lange Primärschlüssel meist ungünstig. Daher ver-

wendet man häufig **künstliche Schlüssel**, bei denen das Attribut extra eingeführt wird, damit man ein identifizierendes Merkmal hat. Im deutschen Sozialversicherungssystem ist das die Sozialversicherungsnummer. In Hochschulen verwendet man oft eine so genannte *Matrikelnummer*, Unternehmen haben Kunden-, Lieferanten-, Rechnungs- und Materialnummern.

Im ERM wird der Primärschlüssel unterstrichen:



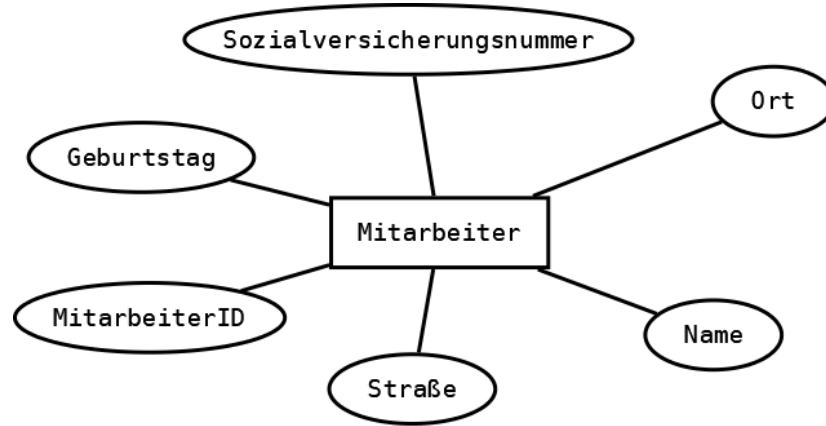
ER-Modell einer Hochschule mit künstlichem Primärschlüssel

Im obigen Beispiel dient die gelbe Färbung des Attributes “Matrikelnummer” lediglich der Hervorhebung der Änderung. Das Datenbankdesign-Team hätte sich auch für jeden der anderen Schlüsselkandidaten entscheiden können. Wobei der zusammengesetzte Schlüsse sehr umfangreich und damit unpraktisch gewesen wäre. Bei der E-Mail-Adresse dachte sich das Team, dass hier Tippfehler vorkommen werden und zur Sozialversicherungsnummer hatte das Immatrikulationsbüro gesagt, dass einige Studierenden vorher nicht in Deutschland gearbeitet haben und daher keine Sozialversicherungsnummer haben.

Vielleicht fragen Sie sich: “Wenn es Primärschlüssel gibt, gibt dann auch **Sekundärschlüssel**?”. Die Antwort lautet “Ja”. Sekundärschlüssel sind Attribute, die zum Suchen in der Datenbank verwendet werden. Sie sind nicht unabdingbar, aber für die Geschwindigkeit der Datenbank von hoher Bedeutung. Will man z. B. in einer Artikeldatenbank oft nach dem Gewicht der Artikel suchen, könnte man das Gewicht als Sekundärschlüssel deklarieren. Da es dabei “nur” um eine Verbesserung der Performanz geht, werden Sekundärschlüssel *nicht* im ERM notiert. Auch müssen sie Sekundärschlüssel in den meisten Datenbankmanagementsystemen nicht eindeutig sein. Sie sind also eigentlich keine Schlüssel. Der Begriff “Indizee” ist daher der bessere.

Aufgabe 10.4: Schlüssel in der Lohnbuchhaltung

Für die Lohnbuchhaltung wurde das folgende ERM entworfen (Ausschnitt):



Lohnbuchhaltung (Ausschnitt)

1. *Ermitteln* Sie Oberschlüssel und Schlüsselkandidaten.
2. *Entscheiden* Sie sich *begründet* für einen der Schlüsselkandidaten als Primärschlüssel.

9.4 ER-Modelle entwickeln

Die Umsetzung der in der realen Welt erkannten relevanten Objekte und Beziehungen in ein Datenbank-Schema erfolgt in mehreren Schritten:

1. **Bildung von Entities:** Wir müssen Einzelobjekte der realen Welt erkennen und zu Entitäten zusammenfassen. (Beispiel: Die Kollegen Fritz Maier und Paul Lehmann und viele weitere werden zur Entität „Angestellte“).
2. **Bestimmung der relevanten Attribute:** Wie jedes Modell ist auch ein ER-Modell (ERM) eine vereinfachte Abbildung der Wirklichkeit, daher werden nicht alle denkbaren Attribute in das ERM aufgenommen werden. (Beispiel: Die Hobbys eines Mitarbeiters sind im Betrieb in der Regel nicht relevant und werden nicht in das ERM aufgenommen.) In Aufgaben in diesem Skript wird oft gefordert, dass nur exemplarische Attribute angegeben werden. Dies dient dann der Arbeitersparnis. Sie müssen dann bei einer Person nicht Name, Vorname, Geburtstag, Straße usw. angeben, sondern beschränken sich auf einige Beispielattribute.
3. **Bildung von Relationships:** Als nächstes müssen wir Einzelbeziehungen zwischen Objekten der realen Welt erkennen und sie zu Relationships zusammen fassen.

Beispiele:

- Der Angestellte Fritz Maier *ist Mitglied* des Qualitätszirkels. ⇒ Relationship „Angestellter ist Mitglied in Gremium“;
- Die Angestellte Paulinna Lehmann *leitet* das Projekt „Verbesserung des Betriebsklimas.“ ⇒ Relationship „Ist Projektleiter“).

Ergänzen Sie hier auch die Kardinalitäten!

9.4.1 Aufgabe 10.5 Buchempfehlung

Für eine Volkshochschule soll ein ER-Modell entwickelt werden, dass die folgenden Anforderungen erfüllt:

Wir haben verschiedene Kurse. Jeder Kurs wird von ein bis zwei Dozenten*innen geleitet.

Die Dozenten*innen geben für jeden ihrer Kurse maximal fünf empfohlene Bücher an.

Entwickeln Sie das geforderte ER-Modell! (Tipp: Sie benötigen drei Entitäten und zwei Relationships.) Beschränken Sie sich bitte jeweils auf fünf exemplarische Attribute.

9.4.2 Aufgabe 10.6 Laufeventmanager

Sie sind Mitarbeiter/in einer eines Softwarehauses und dort u.a. für Datenbanken zuständig. Ihr Unternehmen will ein Content-Management-System zur Verwaltung von Laufsportveranstaltungen entwickeln. Es liegt folgende Beschreibung vor:

Es gibt verschiedene Laufveranstaltungen. Z. B. den Quentiner Stadtlauf des TSV Quentin. Derselbe Veranstalter bietet manchmal aber auch verschiedene Laufveranstaltungen an. So veranstaltet der TSV Quentin auch den Quentiner Adventslauf.

Bei vielen Laufveranstaltungen werden mehrere Distanzen angeboten, z. B. 3, 5 und 10 km, (Halb-)Marathon usw. Es gibt Läufer*innen, die bei einer Veranstaltung mehrere Distanzen absolvieren.

Für jede Distanz erhält jede*r Läufer*in eine Startnummer. Innerhalb der Veranstaltung ist diese Startnummer eindeutig. Die Läufer*innen melden sich zuvor für bestimmte Veranstaltungen und Distanzen an. Die Startnummern werden nicht sofort vergeben, sondern erst, wenn das Startgeld eingegangen ist und der Anmeldeschluss abgelaufen ist.

Die gelaufene Zeit wird gemessen und in die Datenbank eingetragen, um Urkunden und Bestenlisten erstellen zu können. Dabei muss unterschieden werden können, ob ein*e Läufer*in ins Ziel gekommen ist, gestartet aber nicht angekommen ist oder gar nicht erst gestartet ist.

Dabei erfolgt die Auswertung getrennt nach Geschlechtern und Alters-

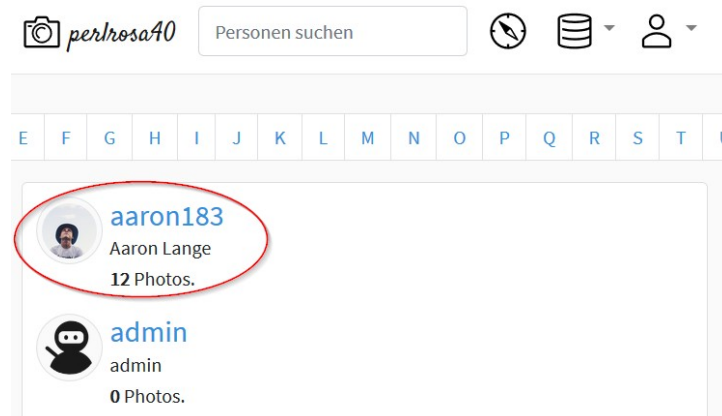
klassen. So gibt es z. B. eine Altersklasse "M90", in der alle männlichen Senioren zwischen 84 und 90 Jahren zusammengefasst sind. In der Altersklasse "W30" sind alle weiblichen Seniorinnen (die heißen wirklich so!) zwischen 26 und 30 Jahren zusammengefasst.

1. *Finden* Sie die nötigen Entitäten für den Laufeventmanager! (Tipp: 5 Entitäten werden es schon sein müssen.)
2. *Bestimmen* Sie relevante (exemplarische) Attribute!
3. *Verbinden* Sie die Entitäten durch passende Relationships!

10 SQL-Abfragen über mehrere Tabellen (JOIN)

Zuletzt hatten Sie in Kapitel 9 mit SQL gearbeitet. Sie hatten die neue Tabelle `photos` erstellt und mit Inhalt gefüllt. Wir kommen nun zurück zu SQL und zu `SELECT` und starten erst einmal in einer Aufwärmübung:

`aaron183` ist einer Ihrer Benutzer. In der Benutzeransicht sehen Sie, dass er 12 Fotos eingestellt hat:



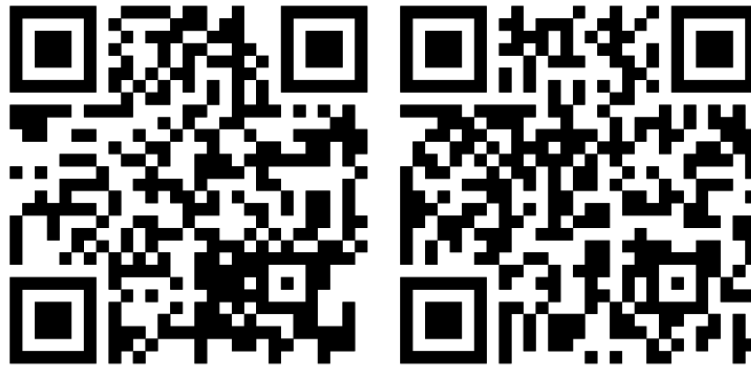
Aaron hat 12 Fotos eingestellt

Wie bekommt InstaHub eigentlich heraus, dass `aaron183` 12 Fotos hochgeladen hat? In der Tabelle `photos` enthält das Feld `user_id` Werte, die in der Tabelle `users` in der Spalte `id` stehen. In Zukunft schreiben wir statt *“Die Spalte `user_id` in der Tabelle `photos`”* einfach kurz *“`photos.user_id`.”*

Aufgabe 11.1: Aarons Fotos

1. *Erklären* Sie das Konzept der Primär- und Fremdschlüssel an Hand des obigen Beispiels!
2. *Erstellen* Sie einen `SELECT`-Befehl der in der Tabelle `users` die `id` des Users mit dem Benutzernamen `aaron183` ermittelt!
3. *Erstellen* Sie einem zweiten `SELECT`-Befehl, der in der Tabelle `photos` die Zahl der von `aaron183` hochgeladenen Fotos ermittelt!

Die folgenden QR-Codes enthalten Lösungsvorschläge für diese Aufgabe:



Lösungsvorschläge für Aufgabe 11.1

10.1 Zwei Tabellen verbinden

Nach diesem Wiedereinstieg in SELECT stellen wir uns natürlich die Frage, ob man immer zwei SELECT-Befehle braucht.

Braucht man nicht. Man kann in der FROM-Klausel auch eine *Liste von Tabellen* angeben:

Aufgabe 11.2 Erster Versuch, zwei Tabellen zu verbinden

Geben Sie den folgenden SQL-Befehl ein:

```
SELECT  COUNT(username)
FROM    users, photos    -- 2 Tabellen in der FROM-Klausel!
WHERE   username="aaron183"
```

1. *Beschreiben* Sie das Ergebnis.
2. *Begründen* Sie welches Ergebnis Sie ungefähr erwartet hätten.
3. *Probieren* Sie einmal folgendes: *Ermitteln* Sie mittels COUNT die Zahl der Datensätze der Tabellen `users` und `photos`. *Ermitteln* Sie dann mittels COUNT die Zahl der Datensätze, die herauskommen, wenn Sie im obigen SELECT die WHERE-Klausel weglassen. In welcher mathematischen Beziehung stehen diese drei Zahlen?

10.2 (INNER) JOIN

Wir müssen SQL sagen, dass nur die zu einander passenden Datensätze aus den beiden Tabellen miteinander kombiniert werden sollen. Das geht so:

```
SELECT  COUNT(username)
FROM    users, photos
WHERE   users.id = photos.user_id    -- Nur die passenden!
        AND username = "aaron183"   -- Nur die von Aaron!
```

In beiden Tabellen gibt es eine Spalte `id`. Woher soll SQL wissen, welche

Spalte wir meinen? Daher muss bei `users.id` die Tabelle angegeben werden. Die Spalte `user_id` gibt es (derzeit) nur einmal daher muss hier ebenso wenig wie bei `username` eigentlich keine Tabelle angegeben werden. In der Abfrage oben wurde die Tabelle `photos` aus optischen Gründen angegeben.

Man kommt recht weit mit dieser Art, Abfragen über mehrere Tabellen zu verfassen. Aber es geht übersichtlicher!

Aufgabe 11.3: JOIN analysieren

Betrachten Sie das folgende SQL-SELECT mit dem neuen Schlüsselwort JOIN (engl. *to join*: sich verbinden):

```
SELECT count(users.id)
FROM    users JOIN photos
        ON (users.id = user_id)
WHERE   username="aaron183"
```

1. *Markieren* Sie in dem obigen SELECT-Befehl alle Stellen, die neue Sprachelemente enthalten!
2. *Begründen* Sie warum die Lösung mit JOIN übersichtlicher ist als die Lösung ohne JOIN. (Wenn Sie noch nicht restlos überzeugt sind: Sie werden gleich noch sehen, dass Sie mit JOIN mächtiger als ohne JOIN sind.)
3. *Probieren* Sie den Befehl aus!

Aufgabe 11.4: JOIN anwenden

Entwickeln Sie jeweils eine SQL-Abfrage mit JOIN!

1. Welche Fotos haben die Mitglieder aus Hamburg hochgeladen?
2. Welches sind die Top-10 der Mitglieder, bezogen auf die Zahl der Fotouploads?
3. Zu jedem Mitglied ist die Zahl der von ihm*ihre hochgeladenen Fotos gesucht.

10.3 Mehr Tabellen für InstaHub! (Tabellenmodelle)

Um komplexere Abfragen behandeln zu können, benötigen wir ein wenig mehr Tabellen.

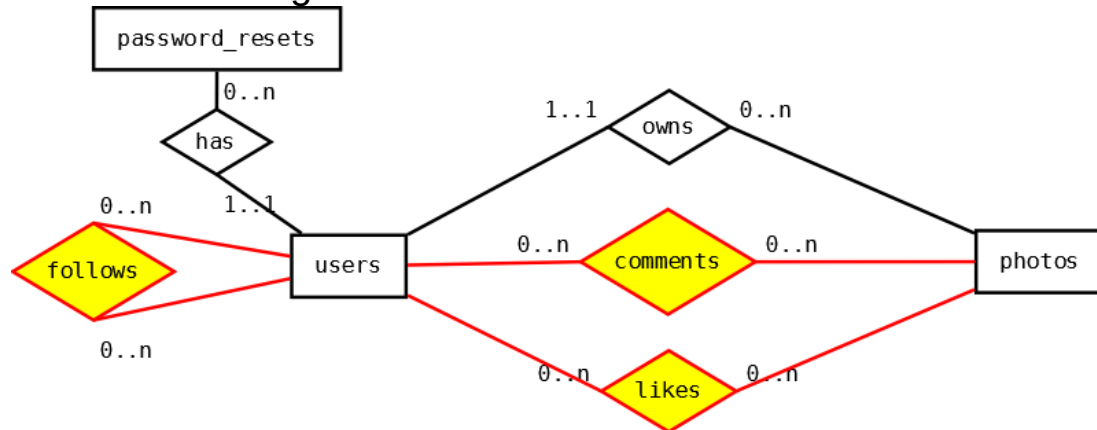
Bisher fehlen InstaHub noch folgende für soziale Netze wichtige Funktionen:

1. Man will anderen Nutzer*innen *folgen* können.

2. Man will Fotos anderer Nutzerinnen liken* können.
 3. Man will Fotos anderer Nutzerinnen kommentieren* können.
- Diese Funktionen bekommt InstaHub jetzt!

Aufgabe 11.5: ER-Modell von InstaHub (2)

Analysieren Sie das folgende ER-Modell für das erweiterte InstaHub!



InstaHub-ERM Version 2

1. Beurteilen Sie die vorgeschlagenen Kardinalitäten!
2. Beurteilen Sie ob das neue Modell die versprochenen neuen Funktionen ermöglicht!

Jeder Datensatz in der Tabelle `users` kann mit jedem Datensatz in der Tabelle `photos` verbunden werden. Und umgekehrt kann jeder Datensatz in der Tabelle `photos` mit jedem Datensatz in der Tabelle `users` verbunden werden. Eine solche Beziehung nennt man auch $n:m$ -Beziehung.

$n:m$ -Beziehungen werden in der Datenbank mit Hilfe einer neuen Tabelle realisiert, die die Primärschlüssel der verbundenen Tabellen als Fremdschlüssel enthält, sie enthält also *zwei Fremdschlüssel*. Die Relation `likes` könnte als Tabelle so aussehen:

<u>user_id</u>	<u>photo_id</u>
15	12
15	14
18	14

In diesem Beispiel hat die*der Benutzer*in mit der `user_id` 15 die Fotos 12 und 14 geliked. Das Foto 14 gefällt auch der*dem Benutzer*in mit der id 18.

Einfacher notiert man Tabellen in der folgenden Form, dem so genannten *Tabellenmodell*:

`likes(^, ^)`

Der Pfeil bedeutet, dass es sich um einen Fremdschlüssel handelt, unterstrichene Attribute sind Teil des Primärschlüssels. Tabellenmodelle sind

nicht genormt, daher werden Sie viele verschiedene Notationen finden.

Anders als ER-Modelle enthalten Tabellenmodelle auch die Fremdschlüssel. Es ist vielfach hilfreich, bei der Umsetzung eines ER-Modells in eine Datenbank das ER-Modell zunächst in ein Tabellenmodell zu überführen.

Aufgabe 11.6: Die Tabellen konstruieren (1)

Entwickeln Sie für die neuen Relationships `comments` und `follows` ein Tabellenmodell!

Aufgabe 11.7: Die Tabellen konstruieren (2)

Entwickeln Sie für jede der drei neuen Tabellen einen SQL-CREATE-Befehl!

Bevor Sie Ihren Befehl abschicken, vergleichen Sie Ihre Lösung unbedingt mit dem Lösungsvorschlag Ihrer Lehrkraft!

Tipp: Gehen Sie dabei wie folgt vor:

1. *Bestimmen* Sie die nötigen Attribute.
2. *Bestimmen* Sie zu jedem Attribut den passenden Datentyp sowie einen Default-Wert und legen Sie fest, ob NULL erlaubt sein soll.
3. *Geben* Sie die *Primärschlüssel* an.
4. *Fügen* Sie die *Fremdschlüssel* hinzu.

Aufgabe 11.8: Die Tabellen konstruieren (3)

Setzen Sie Ihre Create-Befehle aus der vorherigen Aufgabe ab! *Überlegen* Sie zuvor, ob die Reihenfolge Ihrer Befehle eine Rolle spielt!

Aufgabe 11.9: Die Tabellen füllen

Füllen Sie die neuen Tabellen mit den Daten, die Ihre Lehrkraft für Sie bereit gestellt hat. *Überlegen* Sie zuvor, ob die Reihenfolge Ihrer Befehle eine Rolle spielt!

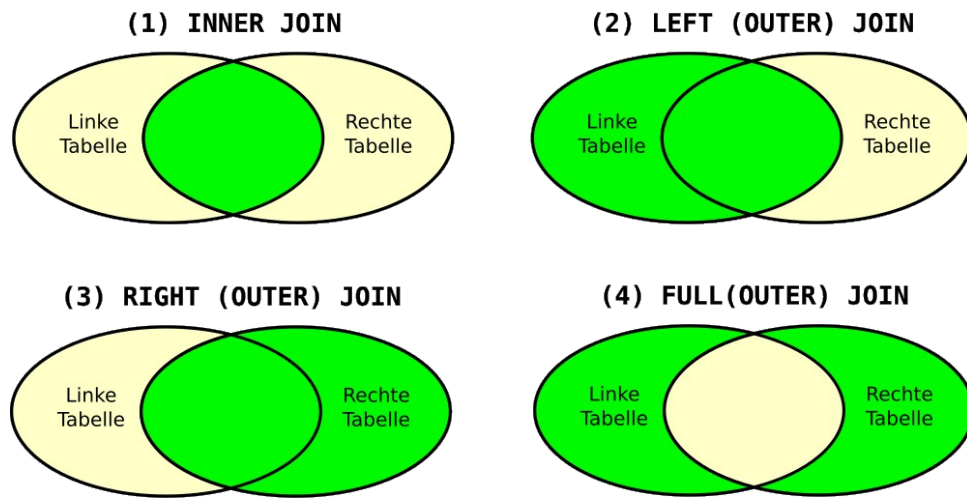
10.4 OUTER JOIN

In Aufgabe 11.4, Nr. 3 hatten Sie zu jedem Mitglied die Zahl seiner Foto-Uploads ermittelt. Sie haben dabei in der JOIN-Klausel etwas geschrieben wie `... ON users.id = photos.userid`. Damit bekommen Sie alle Datensätze, bei denen die Werte in den beiden Spalten übereinstimmen. Was aber ist mit einem Mitglied, dass keine Fotos hochgeladen hat? Da seine `id` in der Spalte `photos.userid` nicht auftaucht, wird es nicht in dem Ergebnis gelistet.

Wir haben also als Ergebnis eine Liste aller Mitglieder mit Foto-Uploads

erzeugt, aber nicht diejenigen Mitglieder die *keine* Foto-Uploads getätigt haben.

In der folgenden Abbildung sehen Sie die vier verschiedenen Formen des JOINS:



Vier JOIN-Arten

1. Beim **INNER JOIN** werden wie gesagt nur diejenigen Datensätze in das Zwischenergebnis übernommen, die einen passenden Datensatz *sowohl* in der linken als auch in der rechten Tabelle haben. In SQL kann das Wort **INNER** weggelassen werden.
2. Beim **LEFT OUTER JOIN** werden alle Datensätze in das Zwischenergebnis übernommen, die in der *linken* Tabelle enthalten sind. Wenn es passende Werte in der *rechten* Tabelle gibt, werden diese aufgenommen, ansonsten wird **NULL** eingetragen. In SQL kann das Wort **OUTER** weggelassen werden.
3. Beim **RIGHT OUTER JOIN** werden alle Datensätze in das Zwischenergebnis übernommen, die in der *rechten* Tabelle enthalten sind. Wenn es passende Werte in der *linken* Tabelle gibt, werden diese aufgenommen, ansonsten wird **NULL** eingetragen. In SQL kann das Wort **OUTER** weggelassen werden.
4. Der **FULL OUTER JOIN** hat in der Praxis nur eine geringe Bedeutung. Hier werden alle Datensätze in das Zwischenergebnis aufgenommen, die in nur in einer der beiden Tabellen stehen. Viele DBMS können keinen **FULL OUTER JOIN**, so auch das von InstaHub.

Auf das Ergebnis des JOINS werden dann die **WHERE**- und die **GROUP BY**-Klausel angewendet.

SQL kennt außerdem noch den **NATURAL JOIN**. Dies ist ein **INNER JOIN**, bei dem das **ON** weggelassen wird und automatisch alle Attribute herangezogen werden, die denselben Namen haben. Der Befehl

```
SELECT *  
FROM   users NATURAL JOIN photos
```

ist identisch mit dem Befehl

```
SELECT *  
FROM   users INNER JOIN photos  
       ON (users.created_at=photos.created_at AND users.updated_at = photos.updated_at)
```

Wenn man mit NATURAL JOIN arbeiten will, muss man also bei der Benennung der Spaltennamen sehr aufpassen.

Aufgabe 11.10 Mehr Aufgaben mit JOIN

Entwickeln Sie jeweils eine SQL-Abfrage mit JOIN! (Tipp: Sie benötigen sowohl INNER als auch OUTER JOIN!)

1. Gesucht sind die Top 10 der am häufigsten gelikten Fotos!
2. Gesucht sind die Top 10 der am häufigsten kommentierten Fotos!
3. Gesucht sind die Top 10 der am häufigsten gefolgten User!
4. Gesucht sind die "Bottom 10", also die am seltensten gelikten Fotos. Ihre Abfrage muss berücksichtigen, dass diese Liste sowohl völlig ungelikte als auch selten gelikte Fotos enthalten kann.
5. Gesucht sind die "Bottom 10", also die am seltensten gefolgten Nutzer*innen, die nach dem 31.12.1990 geboren wurden. Ihre Abfrage muss berücksichtigen, dass diese Liste sowohl Personen ohne jeglichen Follower als auch selten Personen mit wenigen Followern enthalten kann.

11 Komplexe SQL-Abfragen

11.1 Aggregate selektieren

Einige Nutzer*innen haben noch immer keine Fotos auf InstaHub geliket. Diese sollen nun per E-Mail dazu ermuntert werden. Geben Sie eine SQL-Abfrage an, die die E-Mail-Adresse und den Namen der betroffenen Nutzer*innen liefert!

Als ersten Schritt schreiben Sie mit Ihrem derzeitigen Kenntnisstand vielleicht eine Abfrage, die zu allen Nutzer*innen die Zahl aller von ihnen gelikten Nutzer*innen liefert, z. B. etwas wie:

```
SELECT  users.id, users.email, COUNT(_____)
FROM    users _____ JOIN likes
        ON (_____ = likes.user_id)
GROUP BY _____
```

Aufgabe 12.1: Vervollständigen Sie die SQL-Abfrage

Vervollständigen Sie die obige Abfrage!

Kontrollieren Sie Ihre Lösung mit Hilfe des folgenden QR-Codes:



Lösung zu Aufgabe 12.1

Aber wie bekommen Sie nur diejenigen Nutzer*innen angezeigt, bei denen die COUNT-Spalte den Wert 0 enthält? Sie könnten probieren, es mit einer WHERE-Klausel zu versuchen, aber das wird erfolglos sein, weil WHERE die Datensätze selektiert, die danach mittels GROUP BY gruppiert werden und nicht die gruppierten Ergebniszeilen filtert. Hierzu benötigen Sie die Klausel HAVING. Sie funktioniert fast genauso wie die WHERE-Klausel, nur dass sie aus denen durch GROUP BY erstellten aggregierten Zeilen diejenigen auswählt, die einem Kriterium entsprechen. Dies wird in der folgenden Abbildung veranschaulicht:

WHERE sex=m			GROUP BY city		HAVING COUNT(city)<3	
username	sex	city			city	COUNT(city)
max383	m	München			München	2
jen84	f	München			Dresden	3
malte231	d	München			Hamburg	2
karl2	m	München				
karl261	m	Dresden				
brigitte94	f	Dresden				
harry04	m	Dresden				
lyn900	f	Dresden				
max23	m	Dresden				
chanty32	m	Hamburg				
heinbloed	m	Hamburg				
dietlinde9	f	Hamburg				
Hermine2	f	Hamburg				
Dobby80	d	Hamburg				

Zusammenhang zwischen WHERE und HAVING

Aufgabe 12.2: SQL-Abfrage vervollständigen

Vervollständigen Sie Ihre SQL-Abfrage aus Aufgabe 12.1, so dass sie nun nur noch diejenigen Benutzer*innen liefert, die keine Fotos geliket haben!

Aufgabe 12.3: Aggregate selektieren

Erstellen Sie jeweils eine SQL-Abfrage:

1. Welche Fotos wurden bisher gar nicht oder maximal einmal geliket?
2. Welche Nutzer*innen haben noch kein Foto kommentiert?
3. Welche Fotos wurden bisher noch gar nicht oder maximal einmal kommentiert?

Aufgabe 12.4: Einzelgänger*innen finden (schwer)

Einige Nutzer*innen folgen noch immer niemanden auf InstaHub. Diese sollen nun per E-Mail dazu ermuntert werden. Geben Sie eine SQL-Abfrage an, die die E-Mail-Adresse und den Namen der betroffenen Nutzer*innen liefert!

Erstellen Sie eine entsprechende Abfrage!

Hinweise:

- Diese Aufgabe ist deutlich schwerer als man zunächst meint.
- Es ist mehr als ein*e Benutzer*in betroffen.
- Tasten Sie sich am besten schrittweise an die (vielleicht überraschen-

de) Lösung heran, denn `HAVING COUNT(. . .)=0` funktioniert leider nicht.

11.2 Anfragen verschachteln 1: **UNION** und **IN**

Gesucht sind alle Benutzer*innen in Dresden, Hamburg und München. In diesem Abschnitt sehen wir uns verschiedene Lösungsansätze für diese Anforderung an.

11.2.1 Lösung 1: **OR**

Sie können diese Anforderung bereits umsetzen, und zwar mit Hilfe des **OR**-Operators umsetzen:

```
SELECT username, city, name
FROM users
WHERE city = "Leipzig" OR city = "Dresden" OR
      city = "Hamburg"
```

11.2.2 Lösung 2: **IN**

Eine andere Möglichkeit ist der **IN** Operator:

```
SELECT username, city, name
FROM users
WHERE city IN ("Leipzig", "Dresden", "Hamburg")
```

Richtig mächtig wird der **IN**-Operator aber, wenn Sie berücksichtigen, dass in die Klammern ein **SELECT**-Befehl gesetzt werden kann. Das brauchen wir zwar zur Lösung unserer Aufgabe an dieser Stelle nicht, aber es würde wie folgt aussehen:

```
SELECT username, city, name, created_at
FROM users
WHERE city IN (
    SELECT DISTINCT city
    FROM users
    WHERE created_at < "2017-08-31"
)
```

Dieser Befehl liefert die Namen aller Benutzer*innen, die in einer Stadt wohnen, in der jemand wohnt, der vor September 2017 in InstaHub angemeldet war. (Wofür auch immer die Geschäftsleitung das wissen wollte... ;-))

11.2.3 Lösung 3: **UNION**

Sie können die Ergebnisse zweier SQL-Anfragen mit Hilfe des SQL-Befehls **UNION** zusammenfügen:

```
SELECT username, city, name
FROM users
WHERE city = "Leipzig"

UNION

SELECT username, city, name
FROM users
WHERE city = "Dresden"

UNION

SELECT username, city, name
FROM users
WHERE city = "Hamburg"
```

Dabei muss die Zahl der Ergebnisspalten der Teiltabellen übereinstimmen.

Aufgabe 12.5: Unterwegs für die Kriminalpolizei

Auf Grund einer Anzeige durch ein InstaHub-Mitglied ist die Kripo auf Ihren InstaHub aufmerksam geworden. Es besteht der Verdacht, dass auf Ihrem InstaHub volksverhetzende Äußerungen verbreitet werden. Menschen zweifelhafter Gesinnung sollen ihr Unwesen treiben und harmlose Bilder mit Bezügen zu Adolf Hitler kommentieren. Ihnen wird ein richterlicher Beschluss präsentiert, der Sie dazu verpflichtet, folgende Daten herauszugeben:

1. Eine Liste mit allen Benutzer*innen (Name, E-Mail-Adresse, Geburtsdatum), die einschlägig kommentierten Fotos ebenfalls geliket, kommentiert oder hochgeladen haben. Jede*r Nutzer*in darf nur einmalig auf der Liste stehen, es ist nicht wichtig, zu wissen *warum* diese*r Nutzer*in auf die Liste gelangt ist.
2. Eine Liste *aller* Kommentare, die die fraglichen Nutzer*innen abgegeben haben.

11.3 CASE

Für die folgende Aufgabe können Sie den CASE-Operator benutzen. Hier sehen Sie ihn in freier Wildbahn:

```
SELECT url, COUNT(likes.photo_id) AS Anzahl,
CASE
    WHEN COUNT(likes.photo_id)>10 THEN "****"
    WHEN COUNT(likes.photo_id)>5 THEN "***"
    WHEN COUNT(likes.photo_id)>1 THEN "**"
    ELSE "-"
```

```
        END AS Sterne
FROM      photos JOIN likes ON (photos.id = likes.photo_id)
GROUP BY  photos.id
ORDER BY  COUNT(likes.photo_id) DESC
```

Aufgabe 12.7: Texte generieren

1. Die Geschäftsleitung wünscht einen Bericht, in dem hinter jede*m Nutzer*in die Zahl der erhaltenen Likes steht und je nach Geburtsdatum der Text “minderjährig” bzw. “volljährig”.
2. Erweitern Sie die Abfrage aus der vorangegangenen Aufgabe um eine von Ihnen festgelegte Sternebewertung für die Zahl der für alle hochgeladenen Fotos erhaltenen Likes.

11.4 Anfragen verschachteln 2: EXISTS, ANY und ALL und NULL

- Dieser Abschnitt wird noch bearbeitet.

11.5 Zusammenfassende Aufgaben

- Dieser Abschnitt wird noch bearbeitet. # Komplexe CRUD-Befehle
- ** Dieses Kapitel ist in Bearbeitung **
- Integritäten revisited # Big Data
- ** Dieses Kapitel ist in Bearbeitung **

12 Datenschutz

** Dieses Kapitel ist in Bearbeitung **

13 Normalisierung

** Dieses Kapitel ist in Bearbeitung **

14 Datenbankmodelle und Tabellenmodelle

** Dieses Kapitel ist in Bearbeitung **

15 Datenbankmodelle entwickeln

** Dieses Kapitel ist in Bearbeitung **

16 Top 10 der Themen, die nicht behandelt wurden

Datenbanken sind ein sehr weites Gebiet der Informatik. Wir konnten uns nicht um alle Aspekte kümmern. Interessant wäre z. B. noch gewesen:

1. **Office-Datenbanken:** Insbesondere die Datenbank MS Access kann zur Entwicklung kleinerer Datenbankanwendungen verwendet werden.
2. **Integration von Datenbanken in Office-Pakete**, z. B. kann Writer eine einzelne Adressen aus einer Datenbank holen oder einen Serienbrief an alle in einer Abfrage befindlichen Adressen drucken.
3. **PHP und MySQL** sind zwei frei verfügbare Pakete, auf denen viele dynamische Websites basieren. OOo Base kann übrigens auf MySQL-Datenbanken zugreifen.
4. **Tuning von Datenbanken**, z. B. Erstellen von Indizes, mit denen bei großen Datenbanken der Zugriff auf die Datensätze beschleunigt wird, und Berücksichtigung technischer Aspekte bei Entwurfsentscheidungen: Datenbanken werden denormalisiert oder Tabellen geteilt, obwohl dies semantisch unnötig ist.
5. **Relationale Algebra**, die formalisierte Darstellung der Funktion von Datenbanken. Mit ihr kann man sehr präzise beschreiben, wie Datenbanken funktionieren.
6. **Objektorientierte Datenbanken:** Hier müssen nicht alle Datensätze dieselben Attribute haben. Objektorientierte Datenbanken passen gut zur objektorientierten Softwareentwicklung, haben sich aber noch nicht durchgesetzt.
7. **Transaktionssteuerung:** Wie kann man verhindern, dass zwei Reisebüros gleichzeitig den letzten freien Platz im Flugzeug reservieren?
8. **Data Mining:** z. B. Data Warehouses, zentrale Datenbanken von Unternehmen, die fast alle Daten des Unternehmens zusammenfasst) und Data Mining (Auswerten von Datenbeständen um darin Muster zu erkennen, z. B. zur Marktanalyse oder Prognosebildung).
9. **Normalformen:** Diese haben wir angerissen. Es gibt deutlich mehr als drei Normalformen und man kann diese sehr formalisiert beschreiben.

17 SQL- und ER-Modell-Cheat-Sheet

17.1 SQL

17.1.1 Einfache SQL-Abfragen

```
SELECT username AS "Name", city AS "Stadt"
FROM users
WHERE city = "Berlin" AND NOT name LIKE "%Ma_er"
```

- Logische Operatoren: AND, OR, NOT
- Vergleiche: >, <, >=, <=, =, LIKE
- Duplikate in einer Spalte ausschließen: DISTINCT
- Sortieren: ORDER BY
- Zahl der Ausgabezeilen begrenzen: LIMIT

17.1.2 Aggregate

```
SELECT city, MIN(centimeters), MAX(centimeters)
FROM users
GROUP BY city
```

- Aggregatfunktionen: AVG, MIN, MAX, COUNT, SUM
- Aggregate filtern: HAVING

17.1.3 INSERT

```
INSERT
INTO users (
    username, email, password,
    name, bio, gender,
    birthday, city, country,
    centimeters, avatar, role,
    is_active, remember_token, created_at,
    updated_at
)
VALUES (
    'guenther37', 'guenther@instahub.app', '12345',
    'Günther Müller', 'Günther mag Kartoffeln', 'male',
    '2006-06-06 00:00:00', 'Leipzig', 'Deutschland',
    '173', 'avatar.png', 'user',
    '0', NULL, now(),
    now()
)
```

17.1.4 UPDATE

```
UPDATE users
SET country = "Deutschland"
WHERE country = "DDR"
```

17.1.5 DELETE

```
DELETE FROM users
WHERE birthday > "2018-01-01"
```

17.1.6 CREATE

```
CREATE TABLE photos (
  id INT(4) UNSIGNED NOT NULL AUTO_INCREMENT,
  user_id INT(4) UNSIGNED NOT NULL,
  description VARCHAR(255) NOT NULL,
  url VARCHAR(255) NOT NULL,
  created_at TIMESTAMP NOT NULL DEFAULT now(),
  updated_at TIMESTAMP NOT NULL DEFAULT now(),
  PRIMARY KEY (id),
  FOREIGN KEY (user_id)
    REFERENCES users(id)
    ON DELETE CASCADE
)
```

Datentypen: - INT - ganze Zahlen - SMALL INT - ganze Zahlen, meist zwischen -32.768 und +32.767 - TIMESTAMP - Datum und Uhrzeit - VARCHAR(n) - Zeichenkette der variablen Länge *n* - DECIMAL(*p*, *s*) - *p* Stellen insgesamt, *s* davon sind Nachkommastellen - FLOAT - Gleitkommazahl (wie beim Taschenrechner) - CHAR (n) - Zeichenkette fester Länge

17.1.7 JOIN

```
SELECT count(users.id)
FROM users JOIN photos
      ON (users.id = user_id)
WHERE username="aaron113"
```

- LEFT JOIN - Alle Datensätze der linken Tabelle, auch wenn es keinen passenden Datensatz in der rechten Tabelle gibt.
- RIGHT JOIN
- INNER JOIN - identisch mit JOIN
- NATURAL JOIN - identisch mit JOIN, wobei alle Spalten mit identischen Namen in die ON-Klausel übernommen werden.

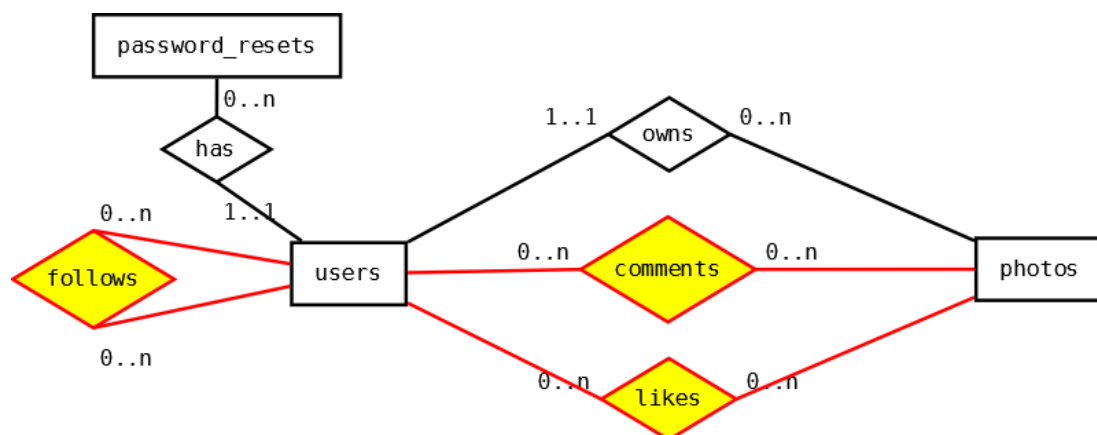
17.1.8 Verschachtelungen

- WHERE city IN ("Dortmund", "Essen", "Köln")
- WHERE city IN (SELECT city FROM users WHERE ...)
- SELECT ... UNION SELECT ...

17.1.9 CASE

```
SELECT    url, COUNT(likes.photo_id) AS Anzahl1,  
          CASE  
            WHEN COUNT(likes.photo_id) > 10 THEN "****"  
            WHEN COUNT(likes.photo_id)>5 THEN "***"  
            WHEN COUNT(likes.photo_id)>1 THEN "**"  
            ELSE "-"  
          END AS Sterne  
FROM      photos JOIN likes ON (photos.id = likes.photo_id)  
GROUP BY  photos.id  
ORDER BY  COUNT(likes.photo_id) DESC
```

17.2 ER-Diagramme



Beispiel für ein ER-Diagramm