**Garlic – Social Media**

3EHIF  
2015/2016

**Team:** Michael Bartl, Maximilian Meyer-Mölleringhof

**Description:**

Garlic is a social network that can be accessed over a local application on your Windows Computer. A new user just needs to sign up with his or her e-mail address and password and can immediately start to write some content. The content that being created is divided into categories (cloves = Knoblauchzehen) which contain articles that users can read and comment on. If a user really likes an article or a comment, he or she can easily upvote it. All upvotes are stored and in the end lead to a certain rank.  
Users can subscribe as well as moderate a clove. Subscribing means that the user gets notifications as soon as there is new content being posted. Moderating means that the user has full access to the clove and can edit / delete / add content that a regular user cannot.  
As Garlic is not the only social media out there, the user can connect his or her other social accounts with it. This is important for sharing content and can also be used for notifications.

Table of Contents

[1. Entity Relationship Diagram – Extended Chen-Notation 3](#_Toc444439720)

[Beschreibung der Attribute 4](#_Toc444439721)

[2. Relationales Modell 4](#_Toc444439722)

[3. Relationales Modell-MySQL Workbench 6](#_Toc444439723)

[4. Data Description Language: Alle Befehle zum Erzeugen der Datenbank 6](#_Toc444439724)

[5. Geschäftsfälle - DML/DQL in der Applikation (beispielhaft, nicht vollständig) 9](#_Toc444439725)

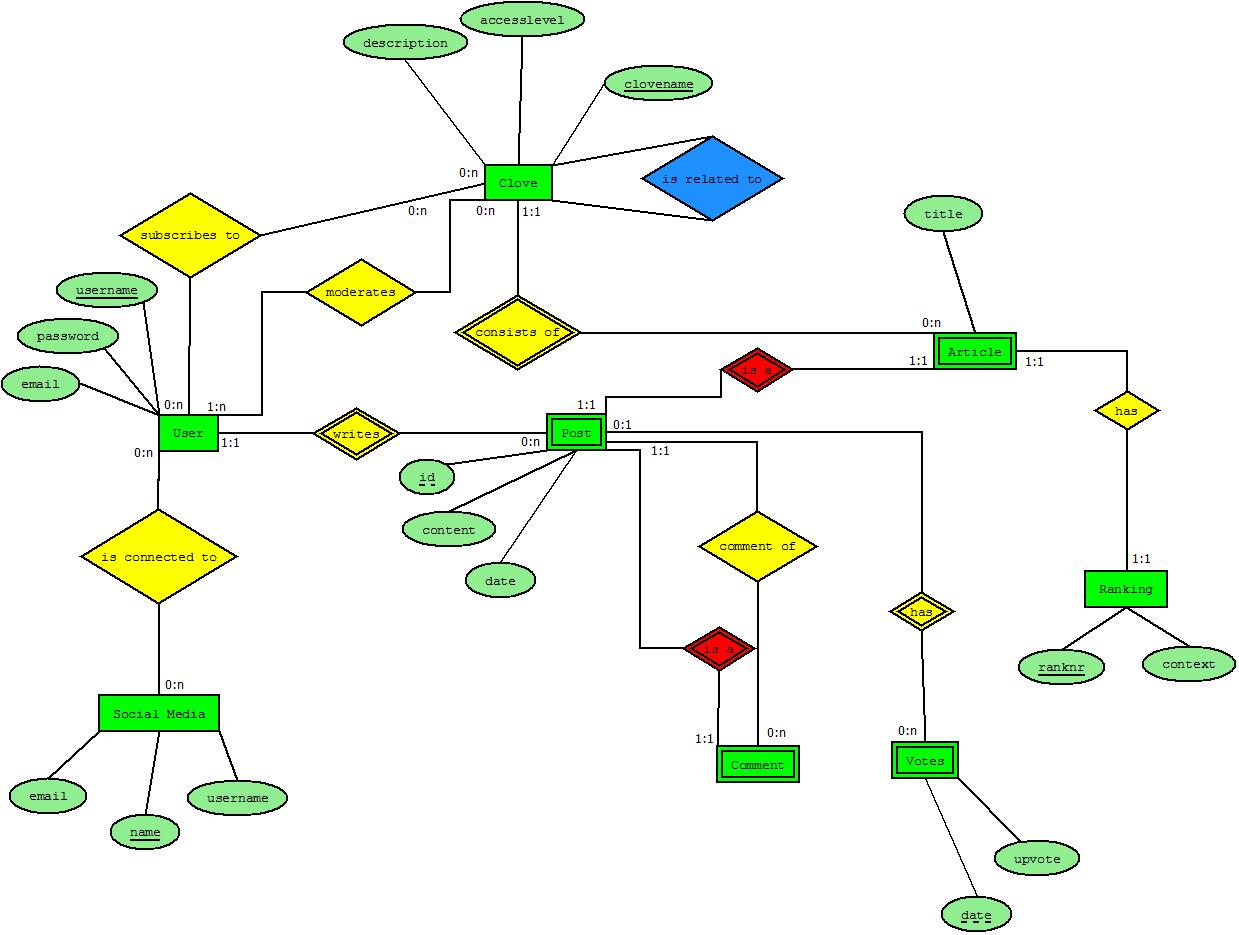
[Einfügen einer neuen Vorlesung 9](#_Toc444439726)

[Einfügen eines neuen Professors 9](#_Toc444439727)

[Alle Vorlesungen aller Professoren 10](#_Toc444439728)

[Alle Vorlesungen eines bestimmten Professors 10](#_Toc444439729)

# **Entity Relationship Diagram – Extended Chen-Notation**



## Beschreibung der Attribute

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity-Set** | **Attribut** | **Beschreibung** | **Datentyp** | **Constraint** |
| Person | PersNr | Eindeutige Personalnummer | Numerisch | Länge 10  Fortlaufend > 0 |
|  | Name | Familienname | Alphanumerisch | Länge 50 |
|  |  |  |  |  |
| Professor | Rang | Einstufung des Professors | Alphanumerisch | Länge 2 |
|  | Raum | Stammraum des Professors | Alphanumerisch | Länge 5 |
|  |  |  |  |  |
| Assistent | Fachgebiet | Welches Fachgebiet: z.B. Numerik | Alphanumerisch | Länge 30 |
|  |  |  |  |  |
| Student | Matrikelnummer | Eindeutige Identifikation eines Studenten. | Numerisch | Länge 8  z. Beispiel:  **90600078**  90 … Jahreszahl  60 … Universität  0078 … fortlaufende  Nummer |
|  | Semester | In welchem Semester befindet sich der Student. | Numerisch | Länge 2 >0 |
| Vorlesung | VorlesungsNr | Eindeutige Nummer der Vorlesung | Numerisch | Länge 7 >0 |
|  | Titel | Bezeichnung der Vorlesung | Alphanumerisch  z.B. Mathematik 1 | Länge 40 |
|  | SemesterWst | Wieviele  Semesterwochenstunden  hat die Vorlesung | Numerisch | Länge 2 >0 |

# **Relationales Modell**

**Bemerkung:** VSA … Vollschlüsselattribut, TSA … Teilschlüsselattribut, FSA … Fremschlüsselattribut **Name der Relation: s\_studenten**

|  |  |  |  |
| --- | --- | --- | --- |
| **Schlüssel** | **DB-Name** | **ERD-Name** | **SQL Datentyp** |
| VSA | s\_matnr | Matrikelnummer | Decimal(8,0), not null |
|  | s\_name | Name | Char(50), not null |
|  | s\_semester | Semester | Decimal(2,0), not null |

**Name der Relation: a\_assistenten**

|  |  |  |  |
| --- | --- | --- | --- |
| **Schlüssel** | **DB-Name** | **ERD-Name** | **SQL Datentyp** |
| VSA | a\_persnr | Personalnummer | Decimal(10,0), not null |
|  | a\_name | Name | Char(50), not null |
|  | a\_fachgebiet | Fachgebiet | Char(30), nullable |
| FSA | a\_p\_boss | Personalnummer | Decimal(10,0), not null |

**Name der Relation: p\_professoren**

|  |  |  |  |
| --- | --- | --- | --- |
| **Schlüssel** | **DB-Name** | **ERD-Name** | **SQL-Datentyp** |
| VSA | p\_persnr | Personalnummer | Decimal(10,0), not null |
|  | p\_name | Name | Char(50), not null |
|  | p\_rang | Rang | Char(2), not null |
|  | p\_raum | Raum | Char(5), not null |

**Name der Relation: v\_vorlesungen**

|  |  |  |  |
| --- | --- | --- | --- |
| **Schlüssel** | **DB-Name** | **ERD-Name** | **SQL-Datentyp** |
| VSA | v\_vorlnr | VorlesungsNr | Decimal(7,0), not null |
|  | v\_titel | Titel | Char(40), not null |
|  | v\_sws | SemesterWst | Decimal(2,0), nullable |
| FSA | v\_p\_gelesenvon | Personalnummer | Decimal(10,0), not null |

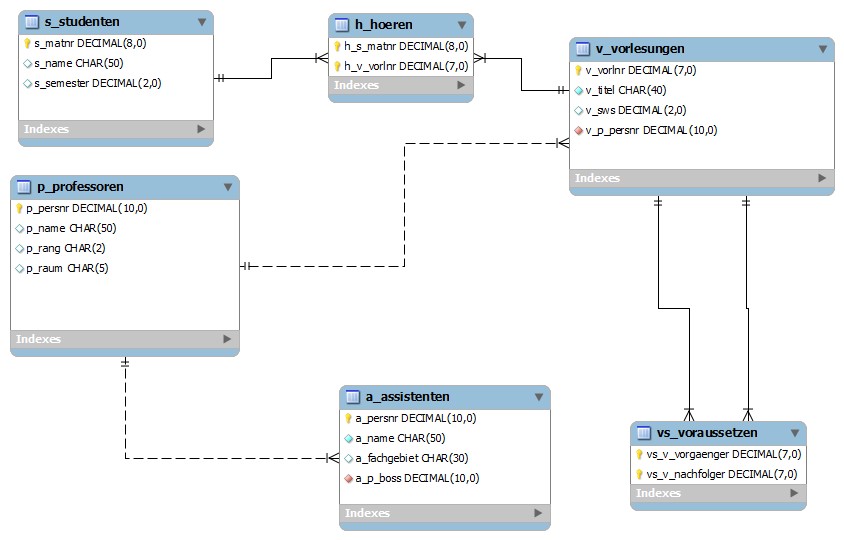
**Name der Relation: h\_hoeren**

|  |  |  |  |
| --- | --- | --- | --- |
| **Schlüssel** | **DB-Name** | **ERD-Name** | **SQL-Datentpy** |
| TSA/FSA | h\_s\_matnr | Matrikelnummer | Decimal(8,0), not null |
| TSA/FSA | h\_v\_vorlnr | VorlesungsNr | Decimal(7,0), not null |

**Name der Relation: vs\_voraussetzen**

|  |  |  |  |
| --- | --- | --- | --- |
| **vs\_voraussetzen** | **DB-Name** | **ERD-Name** | **SQL-Datentyp** |
| TSA/FSA | vs\_v\_vorgaenger | VorlesungsNr | Decimal(7,0), not null |
| TSA/FSA | vs\_v\_nachfolger | VorlesungsNr | Decimal(7,0), not null |

1. Relationales Modell-MySQL Workbench



1. Data Description Language: Alle Befehle zum Erzeugen der Datenbank

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL';

CREATE SCHEMA IF NOT EXISTS `Universitaet` DEFAULT CHARACTER SET latin1 COLLATE latin1\_german2\_ci ;

USE `Universitaet` ;

-- ----------------------------------------------------- -- Table `Universitaet`.`s\_studenten`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `Universitaet`.`s\_studenten` ;

CREATE TABLE IF NOT EXISTS `Universitaet`.`s\_studenten` (

`s\_matnr` DECIMAL(8,0) NOT NULL ,

`s\_name` CHAR(50) NULL ,

`s\_semester` DECIMAL(2,0) NULL ,

PRIMARY KEY (`s\_matnr`) )

ENGINE = InnoDB;

-- ----------------------------------------------------- -- Table `Universitaet`.`p\_professoren`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `Universitaet`.`p\_professoren` ;

CREATE TABLE IF NOT EXISTS `Universitaet`.`p\_professoren` (

`p\_persnr` DECIMAL(10,0) NOT NULL ,

`p\_name` CHAR(50) NULL ,

`p\_rang` CHAR(2) NULL ,

`p\_raum` CHAR(5) NULL ,

PRIMARY KEY (`p\_persnr`) )

ENGINE = InnoDB;

-- ----------------------------------------------------- -- Table `Universitaet`.`v\_vorlesungen`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `Universitaet`.`v\_vorlesungen` ;

CREATE TABLE IF NOT EXISTS `Universitaet`.`v\_vorlesungen` (

`v\_vorlnr` DECIMAL(7,0) NOT NULL ,

`v\_titel` CHAR(40) NOT NULL ,

`v\_sws` DECIMAL(2,0) NULL ,

`v\_p\_persnr` DECIMAL(10,0) NOT NULL ,

PRIMARY KEY (`v\_vorlnr`) ,

INDEX `fk\_v\_vorlesungen\_p\_professoren1` (`v\_p\_persnr` ASC) ,

CONSTRAINT `fk\_v\_vorlesungen\_p\_professoren1`

FOREIGN KEY (`v\_p\_persnr` )

REFERENCES `Universitaet`.`p\_professoren` (`p\_persnr` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- ----------------------------------------------------- -- Table `Universitaet`.`a\_assistenten`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `Universitaet`.`a\_assistenten` ;

CREATE TABLE IF NOT EXISTS `Universitaet`.`a\_assistenten` (

`a\_persnr` DECIMAL(10,0) NOT NULL ,

`a\_name` CHAR(50) NOT NULL , `a\_fachgebiet` CHAR(30) NULL ,

`a\_p\_boss` DECIMAL(10,0) NOT NULL ,

PRIMARY KEY (`a\_persnr`) ,

INDEX `fk\_a\_assistenten\_p\_professoren1` (`a\_p\_boss` ASC) ,

CONSTRAINT `fk\_a\_assistenten\_p\_professoren1`

FOREIGN KEY (`a\_p\_boss` )

REFERENCES `Universitaet`.`p\_professoren` (`p\_persnr` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- ----------------------------------------------------- -- Table `Universitaet`.`h\_hoeren`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `Universitaet`.`h\_hoeren` ;

CREATE TABLE IF NOT EXISTS `Universitaet`.`h\_hoeren` (

`h\_s\_matnr` DECIMAL(8,0) NOT NULL , `h\_v\_vorlnr` DECIMAL(7,0) NOT NULL ,

PRIMARY KEY (`h\_s\_matnr`, `h\_v\_vorlnr`) ,

INDEX `fk\_Student\_has\_Vorlesung\_Vorlesung1` (`h\_v\_vorlnr` ASC) ,

CONSTRAINT `fk\_Student\_has\_Vorlesung\_Student`

FOREIGN KEY (`h\_s\_matnr` )

REFERENCES `Universitaet`.`s\_studenten` (`s\_matnr` )

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Student\_has\_Vorlesung\_Vorlesung1`

FOREIGN KEY (`h\_v\_vorlnr` )

REFERENCES `Universitaet`.`v\_vorlesungen` (`v\_vorlnr` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- ----------------------------------------------------- -- Table `Universitaet`.`vs\_voraussetzen`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `Universitaet`.`vs\_voraussetzen` ;

CREATE TABLE IF NOT EXISTS `Universitaet`.`vs\_voraussetzen` (

`vs\_v\_vorgaenger` DECIMAL(7,0) NOT NULL ,

`vs\_v\_nachfolger` DECIMAL(7,0) NOT NULL ,

PRIMARY KEY (`vs\_v\_vorgaenger`, `vs\_v\_nachfolger`) ,

INDEX `fk\_Vorlesung\_has\_Vorlesung\_Vorlesung2` (`vs\_v\_nachfolger` ASC) ,

CONSTRAINT `fk\_Vorlesung\_has\_Vorlesung\_Vorlesung1`

FOREIGN KEY (`vs\_v\_vorgaenger` )

REFERENCES `Universitaet`.`v\_vorlesungen` (`v\_vorlnr` )

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Vorlesung\_has\_Vorlesung\_Vorlesung2`

FOREIGN KEY (`vs\_v\_nachfolger` )

REFERENCES `Universitaet`.`v\_vorlesungen` (`v\_vorlnr` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

1. Geschäftsfälle - DML/DQL in der Applikation (beispielhaft, nicht vollständig)

Einfügen einer neuen Vorlesung

**Bedingungen:**

Eindeutige Vorlesungsnummer.

Titel der Vorlesung muss angegeben werden.

Angegebener Professor muss vorhanden sein.

**SQL Befehle:**

**…...**

**insert into v\_vorlesungen(v\_vorlnr, v\_titel, v\_sws, v\_p\_persnr) values(vorlesungsnummer, titel, sws, professor);**

Einfügen eines neuen Professors

**Bedingungen:**

Eindeutige Personalnummer. Es darf keinen Professor und keinen Assistenten mit der gleichen Personalnummer geben.

Name des Professors.

Rang des Professors.

Raum.

**SQL Befehle:**

**…..**

**insert into p\_professoren(p\_persnr, p\_name, p\_rang, p\_raum)**

**Values(personalnummer, name, rang, raum);**

Alle Vorlesungen aller Professoren

**select p\_name, v\_vorlnr, v\_titel from v\_vorlesungen join p\_professoren on v\_p\_persnr = p\_persnr**

Alle Vorlesungen eines bestimmten Professors

**select p\_name, v\_vorlnr, v\_titel from v\_vorlesungen join p\_professoren on v\_p\_persnr = p\_persnr where p\_name = ‘Keppler’**