**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 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

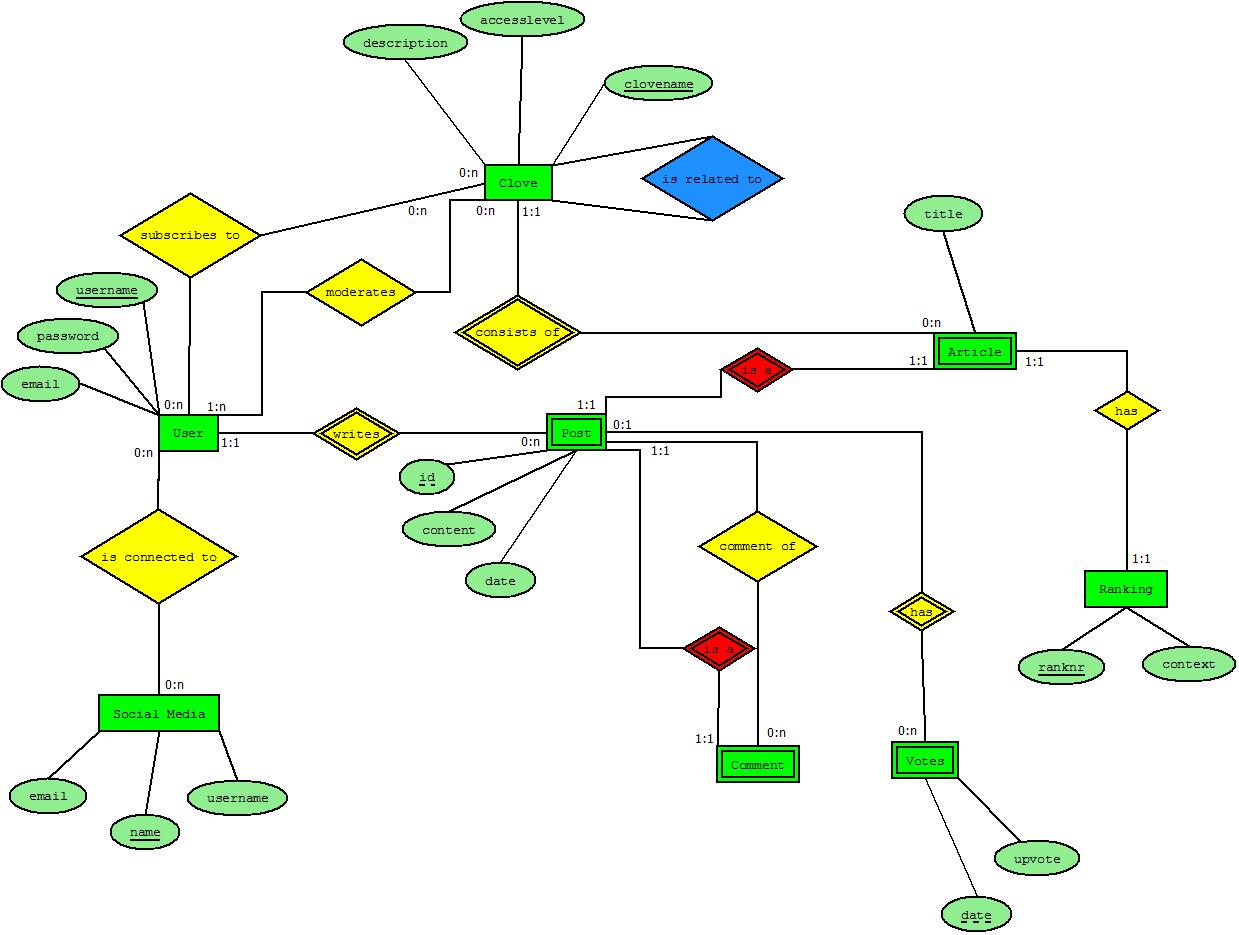
[**1.** **Entity Relationship Diagram – Extended Chen-Notation** 3](#_Toc444675686)

[Description of tables and attributes 4](#_Toc444675687)

[2. Relational Model-MySQL Workbench 5](#_Toc444675688)

[3. Data Description Language: Alle Befehle zum Erzeugen der Datenbank 5](#_Toc444675689)

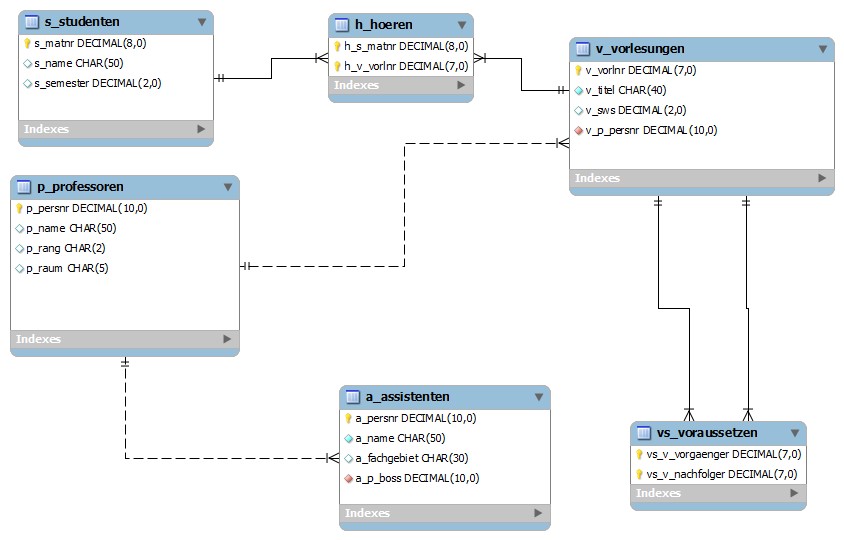
# **Entity Relationship Diagram – Extended Chen-Notation**



## Description of tables and attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity-Set** | **Attributes** | **Description** | **Datatype** | **Constraint** |
| u\_users | username | Identifying username | varchar | Length: 20 |
|  | email | E-Mail of the user | varchar | Length: 50 |
|  | password | Password for the user | varchar | Length: 100 |
| p\_posts | id | Identifying ID for each post | int |  |
|  | content | The posts text | varchar | Length: 10000 |
|  | date | Date when the post has been posted | datetime |  |
| a\_articles | title | The title of the article | varchar | Length: 200 |
| co\_comments |
| c\_cloves | name | Name of the Clove | varchar | Length:50 |
|  | access | Public / private | bool |  |
|  | description | Description of the Clove | varchar | Length:1000 |
| sm\_socialmedias | name | Name of the social media | varchar | Length: 20 |
|  | email | Email the users uses for that social media | varchar | Length: 50 |
|  | username | Username the users uses for that social media | varchar | Length: 50 |
| r\_ranking | ranknr | The rank number of the article | int |  |
|  | context | The context of the rank (Recent, Rising, …) | varchar | Length: 50 |
| v\_votes | upvote | Whether the vote is up or down | bool |  |
|  | date | The date the vote was submitted | datetime |  |

1. Relational Model-MySQL Workbench



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

-- MySQL Script generated by MySQL Workbench

-- 02/27/16 14:28:14

-- Model: New Model Version: 1.0

-- MySQL Workbench Forward Engineering

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,ALLOW\_INVALID\_DATES';

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

-- Schema mydb

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

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

-- Schema mydb

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

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci ;

USE `mydb` ;

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

-- Table `mydb`.`u\_users`

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

CREATE TABLE IF NOT EXISTS `mydb`.`u\_users` (

`u\_username` VARCHAR(20) NOT NULL COMMENT '',

`u\_email` VARCHAR(50) NOT NULL COMMENT '',

`u\_password` VARCHAR(100) NOT NULL COMMENT '',

PRIMARY KEY (`u\_username`) COMMENT '',

UNIQUE INDEX `u\_username\_UNIQUE` (`u\_username` ASC) COMMENT '')

ENGINE = InnoDB;

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

-- Table `mydb`.`p\_posts`

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

CREATE TABLE IF NOT EXISTS `mydb`.`p\_posts` (

`p\_id` INT NOT NULL COMMENT '',

`p\_content` VARCHAR(10000) NOT NULL COMMENT '',

`p\_date` DATETIME NOT NULL COMMENT '',

`p\_u\_username` VARCHAR(20) NOT NULL COMMENT '',

PRIMARY KEY (`p\_id`) COMMENT '',

INDEX `fk\_p\_posts\_u\_users\_idx` (`p\_u\_username` ASC) COMMENT '',

CONSTRAINT `fk\_p\_posts\_u\_users`

FOREIGN KEY (`p\_u\_username`)

REFERENCES `mydb`.`u\_users` (`u\_username`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `mydb`.`co\_comments`

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

CREATE TABLE IF NOT EXISTS `mydb`.`co\_comments` (

`co\_p\_id` INT NOT NULL COMMENT '',

`co\_p\_commentof` INT NOT NULL COMMENT '',

PRIMARY KEY (`co\_p\_id`) COMMENT '',

INDEX `fk\_co\_comments\_p\_posts2\_idx` (`co\_p\_commentof` ASC) COMMENT '',

CONSTRAINT `fk\_co\_comments\_p\_posts1`

FOREIGN KEY (`co\_p\_id`)

REFERENCES `mydb`.`p\_posts` (`p\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_co\_comments\_p\_posts2`

FOREIGN KEY (`co\_p\_commentof`)

REFERENCES `mydb`.`p\_posts` (`p\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `mydb`.`c\_cloves`

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

CREATE TABLE IF NOT EXISTS `mydb`.`c\_cloves` (

`c\_name` VARCHAR(50) NOT NULL COMMENT '',

`c\_access` TINYINT(1) NOT NULL COMMENT '',

`c\_description` VARCHAR(1000) NULL COMMENT '',

PRIMARY KEY (`c\_name`) COMMENT '')

ENGINE = InnoDB;

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

-- Table `mydb`.`sm\_socialmedias`

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

CREATE TABLE IF NOT EXISTS `mydb`.`sm\_socialmedias` (

`sm\_name` VARCHAR(20) NOT NULL COMMENT '',

`sm\_username` VARCHAR(50) NOT NULL COMMENT '',

`sm\_email` VARCHAR(50) NOT NULL COMMENT '',

PRIMARY KEY (`sm\_name`) COMMENT '')

ENGINE = InnoDB;

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

-- Table `mydb`.`r\_rankings`

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

CREATE TABLE IF NOT EXISTS `mydb`.`r\_rankings` (

`r\_rank` INT NOT NULL COMMENT '',

`r\_context` VARCHAR(50) NOT NULL COMMENT '',

PRIMARY KEY (`r\_rank`) COMMENT '')

ENGINE = InnoDB;

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

-- Table `mydb`.`v\_votes`

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

CREATE TABLE IF NOT EXISTS `mydb`.`v\_votes` (

`v\_upvote` TINYINT(1) NOT NULL COMMENT '',

`v\_date` DATETIME NOT NULL COMMENT '',

`v\_p\_id` INT NOT NULL COMMENT '',

PRIMARY KEY (`v\_date`, `v\_p\_id`) COMMENT '',

INDEX `fk\_v\_votes\_p\_posts1\_idx` (`v\_p\_id` ASC) COMMENT '',

CONSTRAINT `fk\_v\_votes\_p\_posts1`

FOREIGN KEY (`v\_p\_id`)

REFERENCES `mydb`.`p\_posts` (`p\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `mydb`.`a\_articles`

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

CREATE TABLE IF NOT EXISTS `mydb`.`a\_articles` (

`a\_p\_id` INT NOT NULL COMMENT '',

`a\_c\_clove` VARCHAR(50) NOT NULL COMMENT '',

`a\_title` VARCHAR(200) NOT NULL COMMENT '',

`a\_r\_rank` INT NOT NULL COMMENT '',

PRIMARY KEY (`a\_p\_id`, `a\_c\_clove`, `a\_r\_rank`) COMMENT '',

INDEX `fk\_a\_articles\_c\_cloves1\_idx` (`a\_c\_clove` ASC) COMMENT '',

INDEX `fk\_a\_articles\_r\_rankings1\_idx` (`a\_r\_rank` ASC) COMMENT '',

CONSTRAINT `fk\_a\_articles\_p\_posts1`

FOREIGN KEY (`a\_p\_id`)

REFERENCES `mydb`.`p\_posts` (`p\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_a\_articles\_c\_cloves1`

FOREIGN KEY (`a\_c\_clove`)

REFERENCES `mydb`.`c\_cloves` (`c\_name`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_a\_articles\_r\_rankings1`

FOREIGN KEY (`a\_r\_rank`)

REFERENCES `mydb`.`r\_rankings` (`r\_rank`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `mydb`.`csm\_connectedsocialmedia`

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

CREATE TABLE IF NOT EXISTS `mydb`.`csm\_connectedsocialmedia` (

`csm\_u\_username` VARCHAR(20) NOT NULL COMMENT '',

`csm\_sm\_name` VARCHAR(20) NOT NULL COMMENT '',

PRIMARY KEY (`csm\_u\_username`, `csm\_sm\_name`) COMMENT '',

INDEX `fk\_u\_users\_has\_sm\_socialmedias\_sm\_socialmedias1\_idx` (`csm\_sm\_name` ASC) COMMENT '',

INDEX `fk\_u\_users\_has\_sm\_socialmedias\_u\_users1\_idx` (`csm\_u\_username` ASC) COMMENT '',

CONSTRAINT `fk\_u\_users\_has\_sm\_socialmedias\_u\_users1`

FOREIGN KEY (`csm\_u\_username`)

REFERENCES `mydb`.`u\_users` (`u\_username`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_u\_users\_has\_sm\_socialmedias\_sm\_socialmedias1`

FOREIGN KEY (`csm\_sm\_name`)

REFERENCES `mydb`.`sm\_socialmedias` (`sm\_name`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `mydb`.`s\_subscription`

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

CREATE TABLE IF NOT EXISTS `mydb`.`s\_subscription` (

`s\_u\_username` VARCHAR(20) NOT NULL COMMENT '',

`s\_c\_clovename` VARCHAR(50) NOT NULL COMMENT '',

PRIMARY KEY (`s\_u\_username`, `s\_c\_clovename`) COMMENT '',

INDEX `fk\_u\_users\_has\_c\_cloves\_c\_cloves1\_idx` (`s\_c\_clovename` ASC) COMMENT '',

INDEX `fk\_u\_users\_has\_c\_cloves\_u\_users1\_idx` (`s\_u\_username` ASC) COMMENT '',

CONSTRAINT `fk\_u\_users\_has\_c\_cloves\_u\_users1`

FOREIGN KEY (`s\_u\_username`)

REFERENCES `mydb`.`u\_users` (`u\_username`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_u\_users\_has\_c\_cloves\_c\_cloves1`

FOREIGN KEY (`s\_c\_clovename`)

REFERENCES `mydb`.`c\_cloves` (`c\_name`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `mydb`.`ad\_admins`

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

CREATE TABLE IF NOT EXISTS `mydb`.`ad\_admins` (

`ad\_u\_username` VARCHAR(20) NOT NULL COMMENT '',

`ad\_c\_clovename` VARCHAR(50) NOT NULL COMMENT '',

PRIMARY KEY (`ad\_u\_username`, `ad\_c\_clovename`) COMMENT '',

INDEX `fk\_u\_users\_has\_c\_cloves\_c\_cloves2\_idx` (`ad\_c\_clovename` ASC) COMMENT '',

INDEX `fk\_u\_users\_has\_c\_cloves\_u\_users2\_idx` (`ad\_u\_username` ASC) COMMENT '',

UNIQUE INDEX `ad\_u\_username\_UNIQUE` (`ad\_u\_username` ASC) COMMENT '',

UNIQUE INDEX `c\_cloves\_c\_name\_UNIQUE` (`ad\_c\_clovename` ASC) COMMENT '',

CONSTRAINT `fk\_u\_users\_has\_c\_cloves\_u\_users2`

FOREIGN KEY (`ad\_u\_username`)

REFERENCES `mydb`.`u\_users` (`u\_username`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_u\_users\_has\_c\_cloves\_c\_cloves2`

FOREIGN KEY (`ad\_c\_clovename`)

REFERENCES `mydb`.`c\_cloves` (`c\_name`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `mydb`.`re\_realtedcloves`

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

CREATE TABLE IF NOT EXISTS `mydb`.`re\_realtedcloves` (

`re\_c\_clovename1` VARCHAR(50) NOT NULL COMMENT '',

`re\_c\_clovename2` VARCHAR(50) NOT NULL COMMENT '',

PRIMARY KEY (`re\_c\_clovename1`, `re\_c\_clovename2`) COMMENT '',

INDEX `fk\_c\_cloves\_has\_c\_cloves\_c\_cloves2\_idx` (`re\_c\_clovename2` ASC) COMMENT '',

INDEX `fk\_c\_cloves\_has\_c\_cloves\_c\_cloves1\_idx` (`re\_c\_clovename1` ASC) COMMENT '',

CONSTRAINT `fk\_c\_cloves\_has\_c\_cloves\_c\_cloves1`

FOREIGN KEY (`re\_c\_clovename1`)

REFERENCES `mydb`.`c\_cloves` (`c\_name`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_c\_cloves\_has\_c\_cloves\_c\_cloves2`

FOREIGN KEY (`re\_c\_clovename2`)

REFERENCES `mydb`.`c\_cloves` (`c\_name`)

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. Views

**# get all votes from articles / comments as well as the total of votes of a user**

drop view if exists vUserVotes;

create view vUserVotes as

select u.u\_username as uu\_user,

(

select count(\*)

from a\_articles a inner join (p\_posts p inner join v\_votes v

on p.p\_id = v.v\_p\_post)

on a.a\_p\_id = p.p\_id

where u.u\_username like p.p\_u\_username

) as uu\_articles\_votes,

(

select count(\*)

from c\_comments c inner join (p\_posts p inner join v\_votes v

on p.p\_id = v.v\_p\_post)

on c.c\_p\_id = p.p\_id

where u.u\_username like p.p\_u\_username

) as uu\_comments\_votes,

(

select count(\*)

from v\_votes v inner join p\_posts p

on v.v\_p\_post = p.p\_id

where u.u\_username like p.p\_u\_username

) as uu\_total\_votes

from u\_users u;

drop view if exists vUserRankings;

create view vUserRankings as

select u.u\_username,

(

select count(\*)

from p\_posts p inner join a\_articles a

on p.p\_id = a.a\_p\_id

where a.a\_r\_rank between 1 and 499 and p.p\_u\_username like u.u\_username

) as ur\_superhot,

(

select count(\*)

from p\_posts p inner join a\_articles a

on p.p\_id = a.a\_p\_id

where a.a\_r\_rank between 500 and 999 and p.p\_u\_username like u.u\_username

) as ur\_hot,

(

select count(\*)

from p\_posts p inner join a\_articles a

on p.p\_id = a.a\_p\_id

where a.a\_r\_rank between 1000 and 1499 and p.p\_u\_username like u.u\_username

) as ur\_rising,

(

select count(\*)

from p\_posts p inner join a\_articles a

on p.p\_id = a.a\_p\_id

where a.a\_r\_rank between 1500 and 2000 and p.p\_u\_username like u.u\_username

) as ur\_upcoming,

(

select count(\*)

from p\_posts p inner join a\_articles a

on p.p\_id = a.a\_p\_id

where p.p\_u\_username like u.u\_username

) as ur\_total

from u\_users u;

**# get every post with the user who created it**

**# as well as all the votes the post has gotten so far**

drop view if exists vPostInfo;

create view vPostInfo as

select p.p\_id as pi\_postID, p.p\_date as pi\_postDate, p.p\_content as pi\_postContent,

(

select u.u\_username

from u\_users u

where u.u\_username = p.p\_u\_username

) as pi\_user,

(

select count(\*)

from v\_votes v

where v.v\_p\_post = p.p\_id

) as pi\_votes,

(

select count(\*)

from c\_comments c

where c.c\_p\_commentOf = p.p\_id

) as pi\_comments,

(

select a.a\_title

from a\_articles a

where a.a\_p\_id = p.p\_id

) as pi\_postTitle

from p\_posts p

order by pi\_postID asc;

**# get the number of subscribers and admins per clove**

drop view if exists vCloveInfo;

create view vCloveInfo as

select c.c\_id as ci\_cloveID, c.c\_name as ci\_cloveName,

(

select count(\*)

from s\_subscriptions s

where s.s\_c\_clove = c.c\_id

) as ci\_subscribers,

(

select count(\*)

from ad\_admins ad

where ad.ad\_c\_clove = c.c\_id

) as ci\_admins,

(

select count(\*)

from a\_articles a

where a.a\_c\_clove = c.c\_id

) as ci\_articles

from c\_clove c;

**# select all the data needed for the homepage of the asp.net client**

drop view if exists vCloveArticles;

create view vCloveArticles as

select a.a\_p\_id, a.a\_title, a.a\_c\_clove, p.p\_u\_username,

(

select c.c\_name

from c\_clove c

where c.c\_id = a.a\_c\_clove

) as cloveName,

(

select c.c\_description

from c\_clove c

where c.c\_id = a.a\_c\_clove

) as cloveDesc,

(

select count(\*)

from c\_comments co

where co.c\_p\_commentOf = p.p\_id

) as commentCount,

(

select count(\*)

from v\_votes v

where v.v\_p\_post = p.p\_id

) as voteCount

from p\_posts p inner join a\_articles a

on p.p\_id = a.a\_p\_id;

1. WPF Projekt

