Centralt kursusstyringssystem for Hjemmeværnets medarbejdere og Hjemmeværnsskolens uddannelse

Casper Kristiansen, s082916

Daniel Martin Shanti, s082941

Kongens Lyngby, Juni 2011

Danmarks Tekniske Universitet
Institut for Informatik og Matematisk Modellering
Bygning 321, 2800 Kongens Lyngby
Tlf +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

FORORD

Denne rapport er skrevet som et bachelor projekt på Danmark Tekniske Universitet DTU. Rapporten beskriver analysen af et kursusstyringssystem til Hjemmeværnet. Læseren bør have et grundlæggende kendskab til software- og databasedesign, for få fuldt udbytte af rapporten. Dog vil de centrale tekniske værktøjer, så som IEEE's model for kravspecifikationer, databasediagrammer og databasenormalisering blive introduceret inden de benyttes, men et forhåndskendskab til teknikkerne vil være en fordel.

Vi vil gerne takke Stig Høgh, for at være vores vejleder på dette projekt. Han har været en stor hjælp under hele processen og har været med til at bringe projektet op på dets nuværende niveau.

Tak til Leder af Personeltjenesten ved Hjemmeværnsdistriktet på Bornholm, Sven-Erik Sørensen, Kompagnichef ved Hærhjemmeværnskompagni Hasle, Karsten Juul Pedersen og Kommandobefalingsmand ved Hærhjemmeværnskompagni Hasle, Flemming Kristiansen, for at være vores kontaktpersoner i Hjemmeværnet og for at have suppleret os med baggrundsviden omkring Hjemmeværnet som organisation samt dets virke. Denne viden har været uvurderlig under projektperioden.

ABSTRACT

We wanted to work with a software project that solves a real problem for a larger group of people. The Course management system for the Home Guard Educational Section was a great candidate for this, primarily because the Home Guard has a very real need for a software system to manage their educations, but also because they have been very helpful in giving us the information that we required to solve this task.

The goal of the project was to specify a system that can replace the manual and time demanding tasks that they are currently stuck with, when they organize their education. This means that we had to specify a system that makes it possible to register information about the members of the Home Guard and the educations that they offer, and also the relations between members and educations.

We handled the problem together with the Home Guard by creating a software requirement specification using the IEEE Software Requirement Specification standard. We then made a systematic analysis of the way the Home Guard works with their current educational systems, and based on this analysis we created a data model that supports the required functionality.

The result of this project is a complete requirement specification for a system that, if implemented, could replace the current workflows of the Home Guard. It is also a data model implemented as an SQL schema for a MySQL database. It is our opinion that by creating the requirement specification and the data model, we have solved the hardest problems related to creating this system, because we have converted a very large and loosely specified set of thoughts and ideas to a concrete and well specified software project.

We realize that what we deliver will not have a significant impact on the Home Guard Educational Section, because we have only specified the system and not implemented it. However, we think that if at a later stage, the Home Guard wishes to implement this system, our work will act as a solid foundation.

INDHOLDSFORTEGNELSE

Forord 3
Abstract 4
Indholdsfortegnelse 5
Resumé 10
Indledning 13
Uformel problembeskrivelse 13
Problemformulering 14
Kravspecifikation 14
Introduktion 14
Kravspecifikations form og formål 14
Virkefelt 15
Ordforklaring 15
Dokumentoversigt 15
Overordnet beskrivelse 16
Produktperspektiv 16
Produktets brugergrupper 17
Produktmiljø 18
Produktfunktioner 18
Gæst 18
Bruger 20
Afdelingsleder 23
Kursusansvarlig 24
Kursusadministrator 27
Brugeradministrator 29
Ikke-funktionelle krav 30
Funktionelle krav 31
Gæst 31
Bruger 32
Afdelingsleder 35
Kursusansvarlig 36
Kursusadministrator 38
Brugeradministrator 40

Adgang til systemet 41
Detaljerede ikke-funktionelle krav 41
Bruger 41
Kursus 42
Funktion 42
Udir 42
Dokument 43
Fagplan 43
Kursusafholdelse 43
Sikkerhed 44
Analyse 45
Research 45
Hvs-info.dk - www 45
Hjemmeværnets Uddannelsesdatabase - Access 45
Data til Hjemmeværnets Uddannelsesdatabase - Excel 46
Nuværende arbejdsgange 46
Medlemmer tilmelder sig kursus 46
Afdelingslederen tilmelder medlemmer til kursus 46
Uddannelsesdatabasen opdateres 47
Vores løsning - kort fortalt 47
Fremgangsmåde 47
hvs-info.dk 47
Access-database 47
hjv.dk 48
Medarbejdere ved Hjemmeværnet 48
Fejl og inkonsistenser i de nuværende løsninger 48
Kursushåndtering 48
Kursusforudsætninger 49
Kurser med fælles indhold 50
Uddannelsesdirektiver 52
Fagplaner og dokumenter 53
Tildeling af Q 54
Uddannelser som ikke giver Q 54
Uddannelser som giver ét Q-nummer 54

Uddannelser som giver mere end et Q-nummer....54
Opdatering af kurser....55
Konsekvens for brugere....56
Konsekvens for andre uddannelser....57
Afholdelse af et kursus....57
Brugere i systemet....58
Brugere og organisationen....58
Brugere og uddannelser....59
Brugere og rettigheder....59
Roller....60
Organisation....61
Organisationens rolle i datamodellen....62
Medarbejderens funktion....63
Autentifikation....63
Entiteter....64
Gennemgang af diagram....64
Uddannelse....64
Organisation....74
Datatyper....77
INT vs SMALL- og TINYINT....77
CHAR vs. VARCHAR....77
TEXT....79
Constraints....79
SQL-eksempel....79
Normalformer....80
Første normalform....80
Anden normalform....81
Tredje normalform....81
Eksempler....82
Diskussion....85
Kurser og uddannelsesdirektiver....85
Konklusion....88
Appendix....89
Punkter i uddannelsdirektiver....89

ITUs standarder for telefonnumre....90
Hjemmeværnets Accessdatabase til registrering af kursusinformation....90
Brugerinformationssiden på **hjv.dk**....91
Fejl og inkonsistenser i **hvs-info.dk**....92
Samlet databasediagram....93
SQL Create-script....94

LISTE OVER FIGURER

USE CASE 1 - Vis kursus 18
USE CASE 2 - Vis uddannelsesdirektiver 19
USE CASE 3 - Log Ind 19
USE CASE 4 - tilmeld til kursus 20
USE CASE 5 - Gennemførte kurser 20
USE CASE 6 - Tilgængelige kurser 21
USE CASE 7 - manglende kurser 21
USE CASE 8 - Rediger personoplysninger 22
USE CASE 9 - status for tilmeldte kurser 22
USE CASE 10 - tilmeld flere til kursus 23
USE CASE 11 - Gennemførte kurser for afdeling 24
USE CASE 12 - opdater kursus 24
USE CASE 13 - Opret kursusinstans 25
USE CASE 14 - opdater kursusinstans 25
USE CASE 15 - registrer bestået eller dumpet kursist 26
USE CASE 16 - Opret kursus 27
USE CASE 17 - opret uddannelsesdirektiv 27
USE CASE 18 - opdater uddannelsesdirektiv 28
USE CASE 19 - udpeg kursusansvarlig 29
USE CASE 20 - opret bruger 29

Figur 1 - Illustration af den overordnede sammenhæng af CCMS 16
Figur 2 - Kursus-entiteten er den stabile del af et kursus. Uddannelsesdirektivet indeholder alle øvrige detaljer om kurset 49
Figur 3 - forudsætninger 49
Figur 4 - Grunduddannelse komplet indeholder andre kurser 50
Figur 5 - relation mellem kurser og uddannelsesdirektiver 51
Figur 6 - hyppigheden af diverse punkter i uddannelsesdirektiver 52
Figur 7 - relationer mellem dokumenter, fagplaner og uddannelsesdirektiver 53
Figur 8 - Grundlæggende skydelærerkursus med Udir 4000 giver 7 forskellige Q'er, hvor de 6 af dem kommer fra andre relaterede uddannelser som er indholdt i uddannelsen, mens det syvende kommer fra uddannelsen selv 55
Figur 9 - Kommandobefalingsmandskursus forudsætter alment befalingsmandskursus, ... 56
Figur 10 - Kursusinstanser af fartøjsmesterskursus 57
Figur 11 - Medarbejder og organisation 59
Figur 12 - forskellen på brugerhierariki (T.V) og rollebaseret rettighedssystem (T.H) 60
Figur 13 - Medarbejder A er knyttet til underafdelingen 'hjemmeværnskompagni 7106 hasle', ... 61
Figur 14 - De to løsningsforslag til afdelingsrelationer 62
Figur 15 - Ansvarsfordeling mellem CCMS og Nemlog-in 63
Figur 16 - databasediagram over uddannelse 65
Figur 17 - Databasediagram over personel 71
Figur 18 - databasediagram over organisation 75
Figur 19 - pladsforbruget for CHAR og VARCHAR 78

Figur 20 - 'Fartøjsmesterkursus, 1435' 82
Figur 21 - bruger på fartøjsmesterkursus 83
Figur 22 - medarbejder har taget kursus A i version y.....86

RESUMÉ

Det har været vores ønske at beskræftige os med et softwareprojekt som løser et virkeligt problem for en større gruppe af mennesker. Her har Hjemmeværnets kursussystem været en oplagt kandidat, primært fordi Hjemmeværnet har et reelt behov for et softwaresystem til at administrere deres uddannelser, men også fordi de har været hjælpsomme og har vist vilje til at give os den information vi har haft behov for.

Det vi har ønsket at opnå med denne opgave, er at specificere et system som kan erstatte de manuelle og tidskrævende arbejdsgange som Hjemmeværnet i øjeblikket er udsat for, i forbindelse med organisationen af deres uddannelse. Det betyder at vi skulle specificere et system med mulighed for at registrere information om Hjemmeværnets medlemmer og de kurser som de udbyder - Herunder også samspillet mellem de to.

Vi greb problemet an ved, i samarbejde med hjemmeværnet, at udfærdige en kravspecifikation efter IEEE's standard for software-kravspecifikationer. Herefter foretog vi en systematisk analyse af Hjemmeværnets arbejdsgange og funktionaliten i deres nuværende systemer, og på baggrund af denne analyse udarbejdede vi en datamodel som understøtter den nødvendige funktionalitet.

Resultatet af projektet er en komplet kravspecifikation for et system som, hvis implementeret, kunne erstatte Hjemmeværnets nuværende arbejdsgange. Derudover er det en datamodel implementeret som et SQL-skema til en MySQL-database. Det er vores opfattelse at vi ved udfærdigelsen af kravspecifikationen og datamodellen, har håndteret de vanskeligste opgaver i forbindelse med udviklingen af dette system, fordi vi har konverteret et stort og meget løst specificeret sæt af tanker og idéer, til et konkret og velspecificeret softwareprojekt.

Vi er klar over at det som vi afleverer i sig selv ikke vil have den store betydning for Hjemmeværnet, fordi vi kun har specificeret systemet, og ikke implementeret det. Vi mener dog at hvis Hjemmeværnet på et senere tidspunkt ønsker en implementation af et kursusstyringssystem, så vil vores arbejde udgøre et solidt fundament.

INDLEDNING

Hjemmeværnet er en militær organisation bestående af 4 niveauer (startende med den højeste instans):

1. Hjemmeværnskommandoen
2. Hjemmeværnsregioner (f.eks. Region Sjælland)
3. Distrikter (f.eks. Det Bornholmske Hjemmeværn)
4. Underafdeling (f.eks. Hjemmeværnskompagni 7106 Hasle)

En central del af det at være i Hjemmeværnet, er den kontinuerlige uddannelse af medlemmerne, der har til formål at ruste dem til de skiftende arbejdsopgaver de bliver sat overfor. Der stilles ligeledes en række krav til hvilke kurser der skal tages, for at besidde en given stilling indenfor Hjemmeværnet. Gennem samtale med personer fra organisationen er vi blevet opmærksomme på at deres nuværende håndtering af medlemmers uddannelse er ude af trit med tiden, fordi den baserer sig på manuelle arbejdsgange og involverer en række uafhængige systemer.

Denne rapport indeholder en specifikation for et kursusstyringssystem som kan erstatte de uhensigtsmæssige arbejdsgange som Hjemmeværnets ansatte i øjeblikket er underlagt.

UFORMEL PROBLEMBESKRIVELSE

Nedenfor findes Hjemmeværnets overordnede beskrivelse af problemet. Denne beskrivelse vil, sammen med en løbende dialog med vores kontaktpersoner, danne fundamentet for den videre problemanalyse.

Hjemmeværnets succes afhænger af hvordan de løser de opgaver som samfundet stiller dem overfor. Dette kræver at man sikrer at medlemmer løbende uddannes og efteruddannes. I Hjemmeværnet håndteres medlemmernes uddannelse af Hjemmeværnsskolen og registreres i et excel-regneark. Regnearket skal benyttes sammen med en Access-database, der først hentes ned fra Hjemmeværnets hjemmeside og gemmes på computeren.

Når databasen bliver opdateret, sendes den nye version ud til de medlemmer der måtte have interesse i den. For at sikre at der altid arbejdes på den nyeste version af databasen, skal man ind og kigge på Hjemmeværnets hjemmeside hver eneste gang man skal arbejde med databasen, hvilket selvsagt ikke er særlig brugervenligt.

Modtager to uafhængige personer den aktuelle version af databasen og hver især tilføjer en ændring, vil de sidde med to forskellige versioner, der herefter skal flettes korrekt sammen, så ingen af to de medarbejderes ændringer går tabt.

Hjemmeværnet ønsker en totalløsning, hvor samtlige informationer gemmes i en enkelt database som opbevares og styres fra et centralt sted, på en måde så mange medlemmer kan arbejde med den samtidig, uden risiko for inkonsistent eller utidssvarende data. Det er nødvendigt at det nye system skal kunne indeholde de kurser som allerede findes på hvs-info.dk, i samme form.

Der vil blive taget udgangspunkt i denne beskrivelse i den analyse som vi vil foretage. Det første trin vil være at tage denne relativt løst formulerede beskrivelse af problemet, og omdanne dette til en formel kravspecifikation, som vi kan basere den videre indgående analyse på.

PROBLEMFORMULERING

Vi vil specificere et system som kan bruges til at registrere Hjemmeværnets medarbejdere og kurser, herunder hvordan de to relaterer sig til hinanden. Det omfatter bl.a. hvilke kurser medarbejdere har gennemført og kan gennemføre, samt detaljer om medarbejdere og kurser. Systemet skal leve op til Hjemmeværnets krav til en erstatning for deres nuværende løsning. Desuden vil vi designe en datamodel som kan understøtte dette system.

KRAVSPECIFIKATION

INTRODUKTION

Denne kravspecifikation er resultatet af en analyse af de brugerbehov som er formuleret af officerer og ledende medarbejder af hjemmeværnet. Følgende personer er vores officielt tilknyttede kilder og kontaktpersoner: Leder af Personeltjenesten ved Hjemmeværnsdistriktet på Bornholm, Sven-Erik Sørensen, Kompagnichef ved Hærhjemmeværnskompagni Hasle, Karsten Juul Pedersen og Kommandobefalingsmand ved Hærhjemmeværnskompagni Hasle, Flemming Kristiansen. Formålet er at specificere et stykke software som giver mulighed for at vedligeholde uddannelsesdirektiver og kursusbeskrivelser, samt registrere forholdet mellem medarbejdere og kurser. De forventede modtagere for denne kravspecifikation er ansatte ved DTU som er tilknyttet dette bachelor-projekt, specifikke personer ansat af hjemmeværnet, samt undertegnede forfattere Casper Kristiansen og Daniel Shanti, som vil stå for en implementation af en datamodel der overholder kravspecifikationen.

KRAVSPECIFIKATIONS FORM OG FORMÅL

Vi har valgt at basere vores kravspecifikation på IEEE's standard 830-1998: IEEE Recommended Practice for Software Requirements Specifications. Denne standard anbefaler nogle retningslinjer for hvordan man opbygger en formel kravspecifikation, der berører alle væsentlige aspekter af en software-løsning. Denne type kravspecifikation fungerer som en aftale mellem kunden som skal modtage løsningen (i vores tilfælde Hjemmeværnet) og os som software-udviklere. Når kravspecifikationen opstår under dialog med kunden kan vi minimere risikoen for misforståelser og behov for at ændre grundlæggende i software-løsningen efter at implementationsfasen er påbegyndt. Dette er væsentligt fordi udviklingstiden for at lave ændringer sent i en udviklingsfase oftest er markant større end hvis den pågældende funktionalitet var planlagt ind i løsningen fra begyndelsen.

En anden væsentlig fordel ved en formel kravspecifikation er at det giver mulighed for at give et mere præcist estimat for udviklingstiden for produktet. Den giver også mulighed for eventuelt at outsource implementations-opgaven til et eksternt firma.

VIRKEFELT

Et centralt kursusstyringssystem for Hjemmeværnets medarbejdere og Hjemmeværnsskolens uddannelse vil blive afviklet som en web applikation, som er hostet på en dedikeret server, eventuelt udliciteret og vedligeholdt af et, for hjemmeværnet, eksternt firma. Systemet skal give mulighed for at nøgleansatte indenfor Hjemmeværnet kan tilføje og opdatere kurser og uddannelsesdirektiver. Derudover skal det være muligt for ansatte og frivillige at tilmelde sig kurser på baggrund af deres funktion og placering indenfor hjemmeværnet. Alle informationer i dette system vil blive lagret i en MySQL-database.

ORDFORKLARING

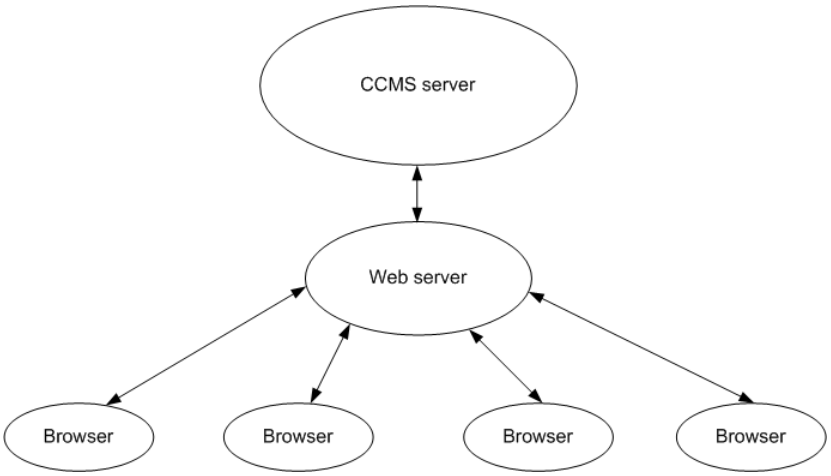
MySQL	Meget udbredt Open Source RDBMS fra Oracle.
RDBMS	Relational Database Management System
IEEE	Institute of Electrical and Electronic Engineers
DTU	Danmarks Tekniske Universitet
CCMS	Centralized Course Management System: Forkortelse for det kursusstyringssystem som beskrives
LAMP	Webapplikations-løsning baseret på Linux, Apache, MySQL, PHP

DOKUMENTOVERSIGT

Den resterende del af kravspecifikationen er inddelt i to overordnede kapitler, hvor det første kapitel beskriver den fulde funktionalitetet af systemet for dets modtagere. Den beskriver alle systemets funktioner og dets overordnede opbygning. Sidste del indeholder en mere teknisk gennemgang af væsentlige detaljer omkring systemet, og skal fungere som en reference for systemets udviklere

OVERORDNET BESKRIVELSE

CCMS skal stilles til rådighed som en web service, hvilket vil sige at et stort antal brugere samtidig kan tilgå og foretage operationer i systemet. Brugerne tilgår systemet gennem deres internet-browser, således at der ikke er behov for at installere bestemte programmer på brugernes computere:



FIGUR 1 - ILLUSTRATION AF DEN OVERORDNEDE SAMMENHÆNG AF CCMS

PRODUKTPERSPEKTIV

CCMS er designet til at erstatte to eksisterende uafhængige systemer, som hjemmeværnet i øjeblikket har i brug. Det ene system som bliver overflødiggjort er hvs-info.dk. Dette system indeholder i øjeblikket en oversigt over alle kurser som Hjemmeværnet stiller til rådighed. Den indeholder også information om Hjemmeværnets medarbejdere, herunder en beskyttet sektion hvor kun medlemmer har adgang. Den indeholder dog ikke funktionalitet som giver mulighed for at medarbejdere kan tilmelde sig kurser, eller oversigter over hvilke kurser den enkelte medarbejder har gennemført.

Det andet system som gøres overflødigt er en decentral Microsoft Access-database som i øjeblikket benyttes til at registrere hvilke kurser de enkelte medarbejdere, indenfor hver region, har gennemført. Denne database benyttes desuden til at afgøre om en medarbejder er kvalificeret til at tage et bestemt kursus, ved at sammenligne de kurser som medarbejderen har taget, med de forudsatte kurser som fremgår på hvs-info.dk.

Systemet vil desuden blive designet på en sådan måde at det på længere sigt vil kunne erstatte andre rutiner indenfor hjemmeværnet, herunder den ikke-digitaliserede kommunikation omkring dokumenter og bekendtgørelser, som i øjeblikket udelukkende foregår per brev. Sidstnævnte opgave er dog ikke en del af dette projekt.

PRODUKTETS BRUGERGRUPPER

Overordnet set kan systemets brugere inddeles i følgende grupper:

Gæster	Personer som besøger webapplikationen uden at identificere sig som brugere. Disse personer har adgang til at læse om kurser og se detaljer om hvilke kurser der forudsætter andre kurser.
Brugere	Menige ansatte og frivillige i Hjemmeværnet. Disse brugere har mulighed for at logge ind og identificere sig overfor systemet. Derudover har de mulighed for at ændre visse personlige detaljer omkring sig selv, og de har mulighed for at se deres egen uddannelsesstatus, herunder hvilke kurser de har gennemført. Endeligt har de mulighed for at tilmelde sig kurser.
Afdelingsledere	Disse brugere har mulighed for at se status for hvilke kurser deres underordnede har gennemført, samt at se og ændre visse personlige detaljer for disse brugere. Desuden har de lov til at tilmelde deres underordnede til kurser. Underordnede anses for at være medarbejdere som tilhører deres egen afdeling, eller afdelinger som ligger lavere i organisationshierarkiet. En afdelingsleder for et distrikt tilhører denne gruppe. Vedkommende betragtes som værende overordnet for alle i samme distrikt, samt alle i underafdelingerne som hører under distriktet. Dog kan der være flere afdelingsledere på en afdeling, og disse anses ikke for at være overordnede eller underordnede i forhold til hinanden.
Kursusansvarlige	En bruger kan være kursusansvarlig for ét eller flere kurser. En kursusansvarlig kan ændre og redigere detaljerne for de kurser som vedkommende er ansvarlig for. Derudover kan han afholde de kurser som han er ansvarlig for, skifte status for de kurser som vedkommende afholder, herunder aflyse og lukke for tilmeldingen til disse kurser. Desuden er det den kursusansvarlige som registrerer om kursister har bestået afsluttede kurser.
Kursusadministratorer	Disse brugere har mulighed for at tilføje, ændre og slette kurser, herunder at oprette nye uddannelsesdirektiver og ændre forudsætningerne for kurser. Kursusadministratorer kan desuden gøre andre brugere til kursusansvarlige.
Brugeradministratorer	Dette er generelt højerestående officerer i Hjemmeværnet som giver nye ansatte adgang til systemet. Brugeradministratorer kan oprette og slette brugere, og kan også tildele og fratage roller fra eksisterende brugere, samt redigere detaljer om de øvrige brugerer.

PRODUKTMILJØ

Selve serveren som hoster CCMS vil være en kraftig dedikeret løsning som har kapacitet til at afvikle applikationen, eksempelvis en LAMP-løsning.

Brugerne vil tilgå webapplikationen via deres browsere, derfor skal der være bred understøttelse for alle moderne browsere, herunder Microsoft Internet Explorer version 7+8+9, Apple Safari 5, Google Chrome 5 samt Mozilla Firefox 3. Løsningen skal udvikles således at den også skal kunne afvikles på mobile browsere, dog vil der ikke blive udviklet applikationer specifikt til mobile enheder.

PRODUKTFUNKTIONER

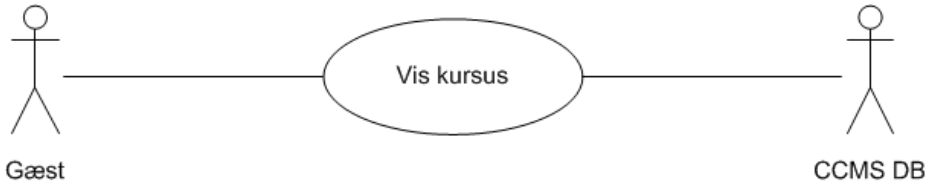
Dette afsnit beskriver den funktionelle kravspecifikation i form af use cases som systemet skal understøtte. Afsnittet indeholder use cases for samtlige af produktets brugergrupper.

Use case diagrammerne indeholder endvidere en række systemer, der ligeledes optræder som aktører:

- CCMS DB: Kursusstyringsystemets database
- NemID DB: NemID's database

GÆST

USE CASE: VIS KURSUS



USE CASE 1 - VIS KURSUS

KORT BESKRIVELSE:
En gæst skal kunne se informationer om et kursus.

TRIN-FOR-TRIN BESKRIVELSE AF FUNKTIONEN:

1. Gæsten vælger "Oversigt over kurser".
2. Gæsten vælger et bestemt kursus.
3. Systemet returnerer alle gemte informationer om kurset.

USE CASE: VIS UDDANNELSESDIREKTIVER FOR KURSUS

USE CASE 2 - VIS UDDANNELSESDIREKTIVER

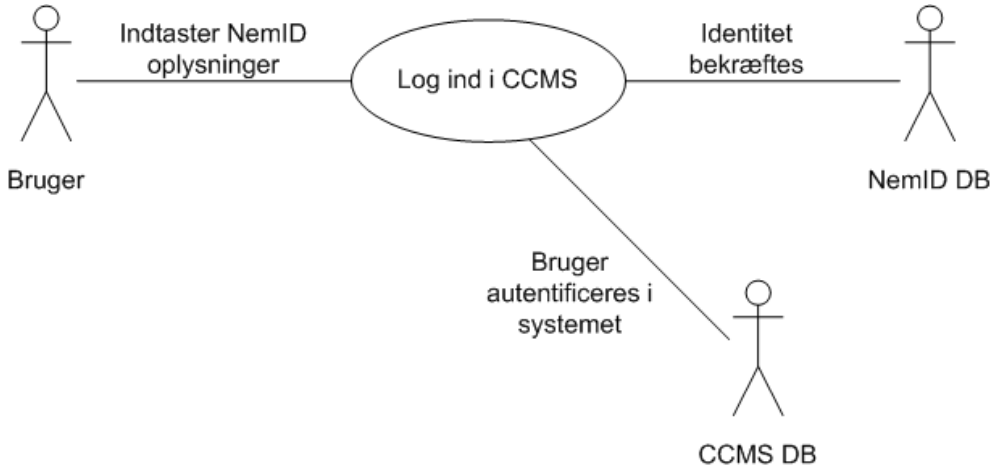
KORT BESKRIVELSE:

En gæst skal kunne se de uddannelsesdirektiver som er gældende for et kursus.

TRIN-FOR-TRIN BESKRIVELSE AF FUNKTIONEN:

1. Gæsten vælger "Oversigt over kurser".
2. Gæsten vælger et bestemt kursus.
3. Gæsten vælger "Vis uddannelsesdirektiver".
4. Systemet returnerer uddannelsesdirektivet for kurset samt eventuelle uddannelsesdirektiver indeholdt af første uddannelsesdirektiv.

USE CASE: LOG IND



USE CASE 3 - LOG IND

KORT BESKRIVELSE:

En gæst skal kunne logge ind.

TRIN-FOR-TRIN BESKRIVELSE AF FUNKTIONEN:

1. Gæsten indtaster MA-nummer og adgangskode i felterne.
2. Gæsten vælger "log ind".
3. Systemet validerer MA-nummer og adgangskode.
4. Hvis gæsten har indtastet ugyldigt MA-nummer eller adgangskode returneres brugeren til forsiden med en besked om at adgangskoden er ugyldig.

5. Hvis gæstens adgangsinformation godkendes, vil denne nu være autentificeret som en bruger i systemet og sendes til sin personlige side som indeholder den pågældende brugers personlige information.

BRUGER

USE CASE: TILMELD TIL KURSUS

USE CASE 4 - TILMELD TIL KURSUS

KORT BESKRIVELSE:
En bruger skal kunne tilmelde sig et bestemt kursus.

TRIN-FOR-TRIN BESKRIVELSE AF FUNKTIONEN:

1. Brugeren er autentificeret i systemet.
2. Brugeren benytter søgefunktionen til at finde et specifikt kursus.
3. Brugeren klikker på "Tilmeld" ud for det ønskede uddannelsessted.
4. Systemet godkender eller afviser tilmeldingen.
5. Resultatet vises for brugeren.

USE CASE: GENNEMFØRTE KURSER

USE CASE 5 - GENNEMFØRTE KURSER

KORT BESKRIVELSE:
En bruger skal kunne se de kurser som han har gennemført.

TRIN-FOR-TRIN BESKRIVELSE AF FUNKTIONEN:

1. Brugeren er autentificeret i systemet.
2. Brugeren vælger "Gennemførte kurser".
3. Systemet laver et opslag baseret på MA-nummer, der indeholder alle Q-numre som brugeren har gennemført.
4. Resultatet vises for brugeren.

USE CASE: TILGÆNGELIGE KURSER

USE CASE 6 - TILGÆNGELIGE KURSER

KORT BESKRIVELSE:

En bruger skal kunne se de kurser som han umiddelbart har adgang til at gennemføre.

TRIN-FOR-TRIN BESKRIVELSE AF FUNKTIONEN:

1. Brugeren er autentificeret i systemet.
2. Brugeren vælger "Kurser med direkte adgang".
3. På basis af MA-nummeret laves et opslag i databasen, som returnerer alle kurser hvor kursuskravene er opfyldt, på baggrund af brugerens gennemførte kurser.
4. Ovenstående mængde præsenteres for brugeren.

USE CASE: MANGLENDE KURSER

USE CASE 7 - MANGLENDE KURSER

KORT BESKRIVELSE:

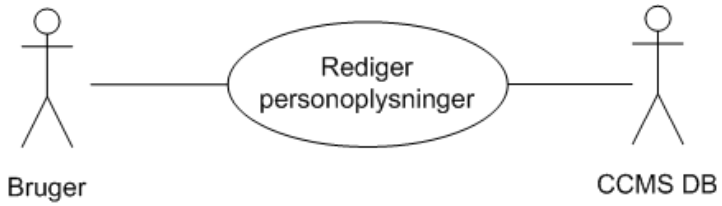
En bruger skal kunne se hvad han mangler for at tilmelde sig et specifikt kursus eller for at påtage sig en specifik funktion

TRIN-FOR-TRIN BESKRIVELSE AF FUNKTIONEN:

1. Brugeren er autentificeret i systemet.
2. Brugeren vælger "Oversigt over kurser" / "Oversigt over funktioner".
3. Brugeren vælger et bestemt kursus/funktion.
4. Systemet laver et opslag efter kurset baseret på Q-nummer. Systemet laver herefter et opslag efter brugerens gennemførte kurser. Der returneres herefter en liste over krævede kurser.

5. Ovenstående mængde præsenteres for brugeren og brugeren kan til- eller fravælge om kurser der allerede er bestået skal fjernes fra listen.

USE CASE: REDIGER PERSONOPLYSNINGER



USE CASE 8 - REDIGER PERSONOPLYSNINGER

KORT BESKRIVELSE:
En bruger skal kunne ændre de oplysninger databasen indeholder om vedkommende.

TRIN-FOR-TRIN BESKRIVELSE AF FUNKTIONEN:

1. Brugeren er autentificeret.
2. Brugeren vælger "Rediger personoplysninger".
3. En formular bliver udfyldt med de oplysninger databasen indeholder om vedkommende, hvorefter brugeren har mulighed for at rette i dem, dog med nogle få undtagelser, herunder MA-nummer og fødselsdato.
4. Brugeren vælger "Gem".
5. Systemet kontrollerer om alle krævede informationer er indtastet.
6. Hvis der mangler at blive udfyldt informationer, eller hvis visse felter er udfyldt med ugyldige værdier. Returneres en side med information om hvad der skal udfyldes igen.
7. Hvis alle krævede detaljer er udfyldt korrekt, opdaterer systemet personoplysningerne.

USE CASE: STATUS FOR TILMELDTE KURSER

USE CASE 9 - STATUS FOR TILMELDTE KURSER

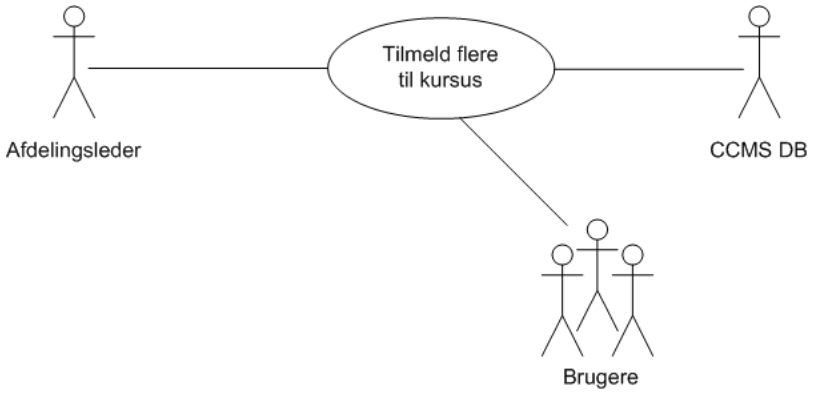
KORT BESKRIVELSE:
En bruger skal kunne se status for alle kurser vedkommende har tilmeldt sig

TRIN-FOR-TRIN BESKRIVELSE AF FUNKTIONEN:

1. Brugeren vælger "Status for tilmeldte kurser".
2. En liste præsenteres med tider og steder hvor et kursus er planlagt, samt aktuel status for kurset, herunder om der er ledige pladser, om det er aflyst osv.

AFDELINGSLEDER

USE CASE: TILMELD FLERE TIL KURSUS



USE CASE 10 - TILMELD FLERE TIL KURSUS

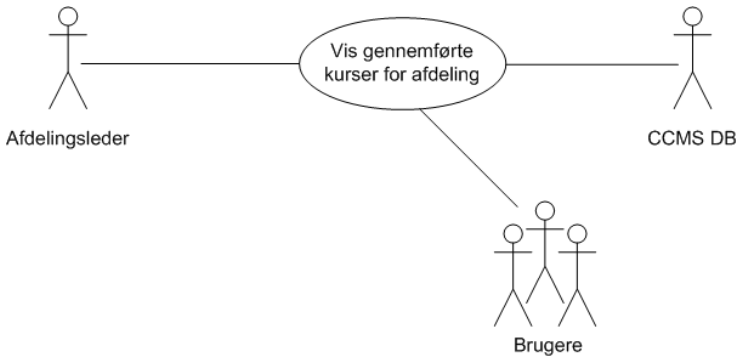
KORT BESKRIVELSE:

En afdelingsleder skal kunne tilmelde brugere i sin afdeling til et kursus.

TRIN-FOR-TRIN BESKRIVELSE AF FUNKTIONEN:

1. Afdelingslederen er autentificeret i systemet.
2. Afdelingslederen benytter søgefunktionen til at finde et specifikt kursus.
3. Afdelingslederen klikker på "Tilmeld flere" ud for det ønskede uddannelsessted.
4. Systemet returnerer en liste over brugere i vedkommendes afdeling, som endnu ikke har gennemført kurset eller er tilmeldt.
5. Afdelingslederen vælger en eller flere brugere og trykker "Tilmeld".
6. Systemet godkender eller afviser tilmeldingen.
7. Resultatet vises for afdelingslederen.

USE CASE: GENNEMFØRTE KURSER FOR AFDELING



USE CASE 11 - GENNEMFØRTE KURSER FOR AFDELING

KORT BESKRIVELSE:

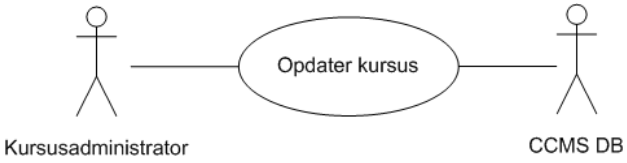
En afdelingsleder skal kunne se de kurser som brugere i vedkommendes afdeling har gennemført.

TRIN-FOR-TRIN BESKRIVELSE AF FUNKTIONEN:

1. Afdelingslederen er autentificeret i systemet.
2. Afdelingslederen vælger "Gennemførte kurser for afdeling".
3. Systemet laver et opslag baseret på MA-nummer for alle brugere i afdelingen, der indeholder alle Q-numre som brugerne har gennemført.
4. Resultatet vises for afdelingslederen.

KURSUSANSVARLIG

USE CASE: OPDATER KURSUS



USE CASE 12 - OPDATER KURSUS

KORT BESKRIVELSE

En kursusadministratoren skal kunne ændre informationerne om et eksisterende kursus

TRIN-FOR-TRIN BESKRIVELSE:

1. Kursusadministratoren er autentificeret i systemet og har ret til at opdatere et kursus.
2. Kursusadministratoren vælger "Oversigt over kurser".
3. Kursusadministratoren vælger et bestemt kursus.
4. Kursusadministratoren vælger "Rediger kursus".

5. Systemet returnerer de nuværende detaljer for kurset på en formular hvor kursusadministratoren har mulighed for at ændre enkelte detaljer.
6. Kursusadministratoren vælger "Gem".
7. Hvis der mangler at blive udfyldt informationer, eller hvis visse felter er udfyldt med ugyldige værdier, returneres en side med information om hvad der skal udfyldes igen.
8. Hvis alle krævede detaljer er udfyldt korrekt, gemmer systemet det ændrede kursus i databasen.

USE CASE: OPRET KURSUSINSTANS

USE CASE 13 - OPRET KURSUSINSTANS

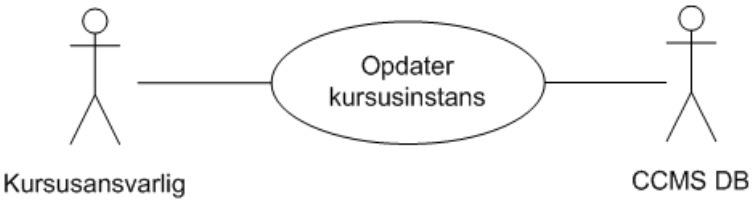
KORT BESKRIVELSE:

En kursusansvarlig skal kunne oprette en kursusinstans

TRIN-FOR-TRIN BESKRIVELSE:

1. Kursusansvarlig er autentificeret i systemet og har ret til at oprette en kursusinstans.
2. Kursusansvarlig vælger "Opret kursusinstans".
3. Kursusansvarlig sendes til en side hvor det er muligt at vælge hvilket kursus der skal oprettes en kursusinstans for.
4. Tid og sted for afholdelsen af kurset indtastes.
5. Kursusansvarlig trykker "Opret".
6. Systemet kontrollerer om alle informationer er indtastet.
7. Hvis der mangler at blive udfyldt informationer, eller hvis visse felter er udfyldt med ugyldige værdier, returneres en side med information om hvad der skal udfyldes igen.
8. Hvis alle krævede detaljer er udfyldt korrekt, opretter systemet den nye kursusinstans i databasen.

USE CASE: OPDATER KURSUSINSTANS



USE CASE 14 - OPDATER KURSUSINSTANS

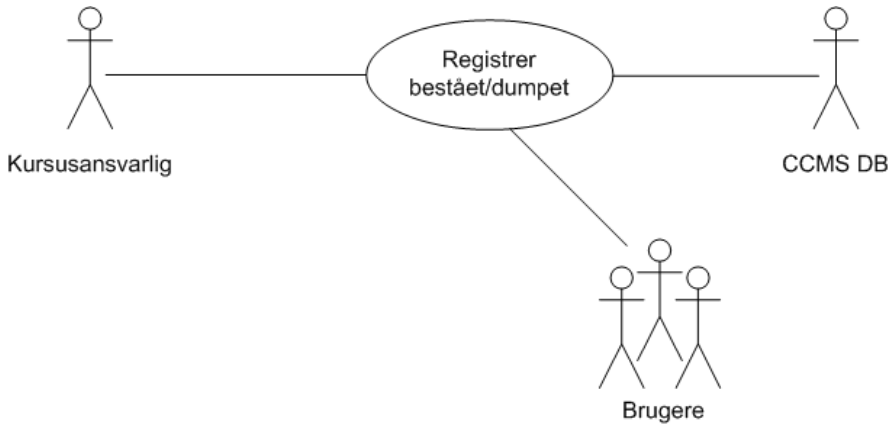
KORT BESKRIVELSE

En kursusansvarlig skal kunne ændre informationerne om en eksisterende kursusinstans

TRIN-FOR-TRIN BESKRIVELSE:

1. Kursusansvarlig er autentificeret i systemet og har ret til at opdatere et uddannelsesdirektiv.
2. Kursusansvarlig vælger "Oversigt over kursusinstanser".
3. Kursusansvarlig vælger en kursusinstans.
4. Kursusansvarlig vælger "Opdater kursusinstans".
5. Systemet returnerer de nuværende detaljer for kursusinstansen og udfylder en formular hvor kursusansvarlig har mulighed for at ændre enkelte detaljer.
6. Kursusansvarlig vælger "Gem".
7. Hvis der mangler at blive udfyldt informationer, eller hvis visse felter er udfyldt med ugyldige værdier, returneres en side med information om hvad der skal udfyldes igen.
8. Hvis alle krævede detaljer er udfyldt korrekt gemmes ændringerne og kursusansvarlig viderestilles til siden for kursusinstansen med de nylige ændringer.

USE CASE: REGISTRER BESTÅET ELLER DUMPET KURSIST



USE CASE 15 - REGISTRER BESTÅET ELLER DUMPET KURSIST

KORT BESKRIVELSE

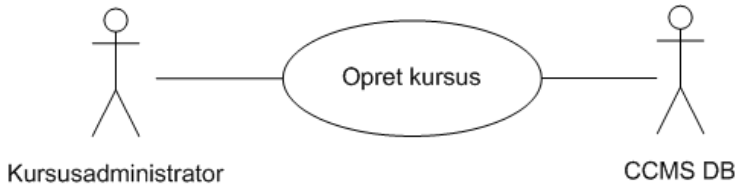
En kursusansvarlig skal kunne registrere hvorvidt en kursist har bestået eller dumpet et kursus

TRIN-FOR-TRIN BESKRIVELSE:

1. Kursusansvarlig er autentificeret i systemet og har ret til at opdatere et uddannelsesdirektiv.
2. Kursusansvarlig vælger "Oversigt over kursusinstanser".
3. Kursusansvarlig vælger en kursusinstans.
4. Kursusansvarlig vælger "Registrer kursister".
5. En liste over kursister præsenteres for kursusansvarlig. Ud for hver kursist er et afkrydsningsfelt der angiver bestået/dumpet.
6. Kursusansvarlig krydser af i feltet, for alle kursister der har bestået og trykker "Gem".
7. Kursusansvarlig viderestilles til en side der viser de nylige ændringer.

KURSUSADMINISTRATOR

USE CASE: OPRET KURSUS



USE CASE 16 - OPRET KURSUS

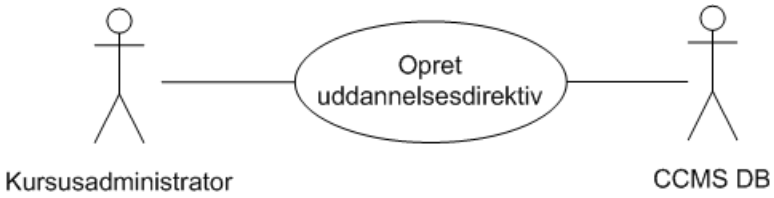
KORT BESKRIVELSE:

En kursusadministratoren skal kunne oprette et kursus

TRIN-FOR-TRIN BESKRIVELSE:

1. Kursusadministratoren er autentificeret i systemet og har ret til at oprette et kursus.
2. Kursusadministratoren vælger "Opret kursus".
3. Kursusadministratoren sendes til en formular hvor navnet for kort og lang titel for kurset indtastes.
4. Kursusadministratoren vælger "Gem".
5. Systemet kontrollerer om alle informationer er indtastet.
6. Hvis der mangler at blive udfyldt informationer, eller hvis visse felter er udfyldt med ugyldige værdier, returneres en side med information om hvad der skal udfyldes igen.
7. Hvis alle krævede detaljer er udfyldt korrekt, opretter systemet det nye kursus i databasen.
8. Kursusadministratoren sendes nu til en side hvor der oprettes et uddannelses-direktiv for kurset.

USE CASE: OPRET UDDANNELSESDIREKTIV



USE CASE 17 - OPRET UDDANNELSESDIREKTIV

KORT BESKRIVELSE:

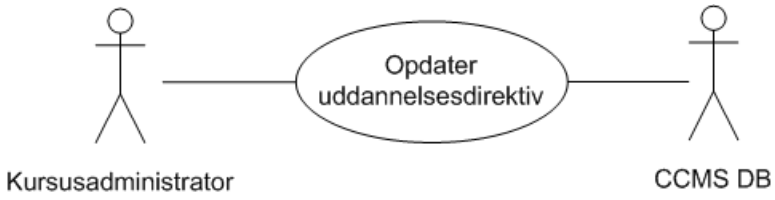
En kursusansvarlig skal kunne oprette et kursus

TRIN-FOR-TRIN BESKRIVELSE:

1. Kursusadministratoren er autentificeret i systemet og har ret til at oprette et uddannelsesdirektiv.
2. Kursusadministratoren vælger "Opret uddannelsesdirektiv".
3. Kursusadministratoren sendes til en formular hvor UDIR indtastes sammen med informationer om uddannelsdirektivet såsom mål, formål, indhold osv.
4. Kursusadministratoren vælger "Gem".
5. Systemet kontrollerer om alle krævede informationer er indtastet.

6. Hvis der mangler at blive udfyldt informationer, eller hvis visse felter er udfyldt med ugyldige værdier, returneres en side med information om hvad der skal udfyldes igen.
7. Hvis alle krævede detaljer er udfyldt korrekt, opretter systemet det nye uddannelsesdirektiv i databasen.
8. Kursusadministratoren sendes nu til en side hvor det er muligt at oprette et kursus for uddannelsesdirektivet.

USE CASE: OPDATER UDDANNELSESDIREKTIV



USE CASE 18 - OPDATER UDDANNELSESDIREKTIV

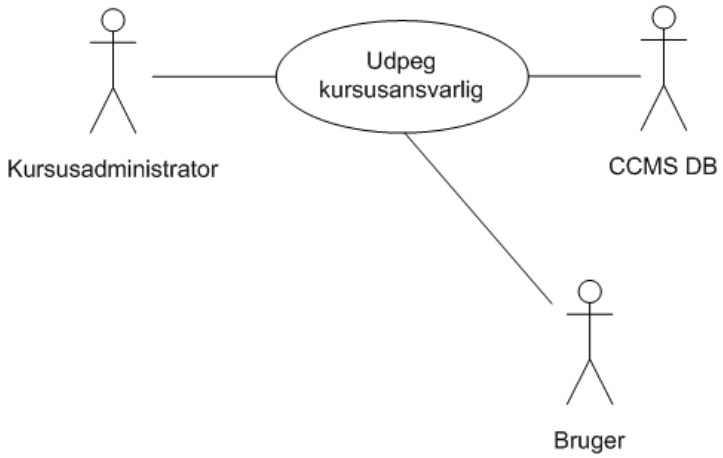
KORT BESKRIVELSE

En kursusansvarlig skal kunne ændre informationerne om et eksisterende kursus

TRIN-FOR-TRIN BESKRIVELSE:

1. Kursusadministratoren er autentificeret i systemet og har ret til at opdatere et uddannelsesdirektiv.
2. Kursusadministratoren vælger "Oversigt over uddannelsesdirektiver".
3. Kursusadministratoren vælger et bestemt uddannelsesdirektiv.
4. Kursusadministratoren vælger "Rediger uddannelsesdirektiv".
5. Systemet returnerer de nuværende detaljer for kurset på en formular hvor kursusadministratoren har mulighed for at ændre enkelte detaljer.
6. Kursusadministratoren vælger "Gem".
7. Hvis der mangler at blive udfyldt informationer, eller hvis visse felter er udfyldt med ugyldige værdier, returneres en side med information om hvad der skal udfyldes igen.
8. Hvis alle krævede detaljer er udfyldt korrekt, bliver kursusadministratoren spurgt om ændringerne er radikale og om der er behov for at uddannelsesdirektivet oprettes som et nyt.
9. Vælger kursusadministratoren at sige ja til dette, vil det gamle uddannelsesdirektiv forblive intakt, mens et nyt vil blive oprettet, med et andet nummer. Kurser der peger på det gamle uddannelsesdirektiv vil blive ændret, så de peger på det nye.

USE CASE: UDPEG KURSUSANSVARLIG



USE CASE 19 - UDPEG KURSUSANSVARLIG

KORT BESKRIVELSE

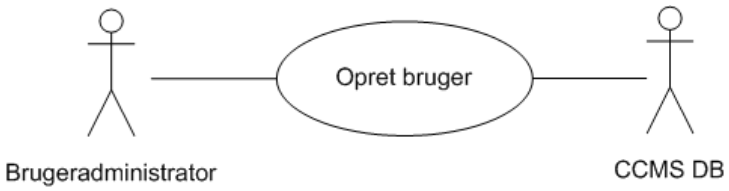
En kursusadministrator skal kunne udpege en eller flere kursusansvarlige for et kursus

TRIN-FOR-TRIN BESKRIVELSE:

1. Kursusadministratoren finder kursus i "Oversigt over kurser" eller ved søgning.
2. Kursusadministratoren vælger "Udpeg kursusansvarlig"
3. Systemet returnerer en liste over potentielle kursusansvarlige med søgemuligheder blandt disse.
4. Kursusadministratoren vælger en eller flere bruger og trykker "Gem".
5. Systemet tilføjer brugeren til listen over kursusansvarlige på det pågældende kursus og returnerer kursusadministratoren til kursussiden.

BRUGERADMINISTRATOR

USE CASE: OPRET BRUGER



USE CASE 20 - OPRET BRUGER

KORT BESKRIVELSE:

En brugeradministrator skal kunne oprette en bruger

TRIN-FOR-TRIN BESKRIVELSE:

1. Brugeradministratoren udfylder personoplysninger om brugeren og trykker "Opret".
2. Serveren verificerer det indtastede og giver mulighed for at rette eventuelle fejl.

3. Er alle felter gyldige, oprettes brugeren med en autogenereret adgangskode, der sendes til brugerens email.
4. Systemet returnerer en liste over tilgængelige roller og præsenterer dem for brugeradministratoren.
5. Brugeradministratoren tildeler eventuelle roller til brugeren og klikker "Gem".

BRUGERKARAKTERISTIK

Den almindelige bruger af systemet forventes at kunne navigere rundt på en hjemmeside og være i stand til at benytte en søgemaskine. Brugeren forventes at kende sin adgangskode, samt at kunne benytte formularen til at logge ind.

Brugere med ret til at oprette kurser og uddannelsesdirektiver bør være i stand til at imødekomme krav til formateringen af felterne i en formular. Således vil et ugyldigt input resultere i en fejlmeddelelse der forklarer det korrekte format, som brugeren herefter må tilpasse input efter.

IKKE-FUNKTIONELLE KRAV

Systemet tænkes udviklet som en rig web-applikation på en webserver med en højhastigheds internetforbindelse, baseret på Extended Google Web Toolkit. Denne applikation vil drage fordel af Windows-miljøets velkendte opbygning hos både erfarne og ikke-erfarne computerbrugere og vil af udseende derfor minde om dette. Her tænkes der specielt på følgende opdeling, der i hjemmeside øjemed er lidt utraditionel:

Skrivebord	Genveje til forskellige programmer Åbne programmer vises her
Startlinje	Startmenu Hurtig-start af hyppigt brugte programmer Liste over åbne programmer, både aktive og minimerede
Åbne programmer	Vindue med knapper til at minimere, maksimere samt lukke programmet Mulighed for at trække vinduet rundt i browseren

Databasedelen kører på en MySQL server, der som minimum ligger på samme fysiske lokation som webserveren, optimalt også på samme fysiske server.

For brugeren er den eneste forudsætning for at benytte systemet, at der bruges en browser med fuld understøttelse for Javascript herunder AJAX gennem jQuery.

FUNKTIONELLE KRAV

GÆST

USE CASE: VIS KURSUS

Use case	**Vis kursus**
Aktivering	Gæsten klikker på linket "Oversigt over kurser"
Prækondition	Ingen
Primær sti	1. Gæsten sender en forespørgsel til serveren omkring udbudte kurser. 2. Serveren returnerer alle kurser og præsenterer det som en liste for gæsten. 3. Gæsten vælger et specifikt kursus. 4. Serveren returnerer alle gemte informationer om kurset 5. Informationerne præsenteres for gæsten.
Alternative stier	Ingen
Postkondition	Gæsten kan se en oversigt over kurset
Særtilfælde	Ingen
Andet	Intet

USE CASE: VIS UDDANNELSESDIREKTIVER FOR KURSUS

Use case	**Vis uddannelsesdirektiver for kursus**
Aktivering	Gæsten klikker på linket "Vis uddannelsesdirektiver"
Prækondition	Ingen
Primær sti	1. Gæsten sender en forespørgsel til serveren omkring kursets uddannelsesdirektiver. 2. På basis af kursus ID, laves et opslag i databasen der returnerer kursets uddannelsesdirektiv og alle uddannelsesdirektiver dette måtte indeholde. 3. Resultatet præsenteres for gæsten som en liste
Alternative stier	Ingen
Postkondition	Gæsten kan se en liste over uddannelsesdirektiver der er indeholdt i kurset
Særtilfælde	Indeholder kurset ingen uddannelsesdirektiver, vil listen være tom
Andet	Intet

USE CASE: LOG IND

Use case	**Log ind**
Aktivering	Gæsten udfylder formular felterne og klikker på "Log ind"-knappen.
Prækondition	Gæsten har et gyldigt brugernavn og adgangskode
Primær sti	1. Gæsten befinder sig på en side med log ind formularen.

	2. Formularen udfyldes brugeroplysninger og der trykkes på “Log ind”. 3. Serveren checker om et felt var tomt. 4. Hvis ingen felter var tomme sendes det indtastede til serveren og bliver verificeret. 5. Hvis det indtastede er gyldigt genindlæses siden, og vedkommende er nu autentificeret som en bruger.
Alternative stier	4. Hvis et felt var tomt, vises formularen på ny, med en fejlmeddelelse der indikerer dette i skridt 2 6. Er indtastningen ugyldig, genindlæses siden og en fejlmeddelelse vil indikere dette i skridt 2
Postkondition	Brugeren er autentificeret i systemet og kan foretage handlinger i forhold til brugerens rettighedsniveau.
Særtilfælde	Hvis gæsten ikke trykker på log ind, vil formulardataene blive glemt, når gæsten klikker videre på en ny side. Der vil ikke ske ændringer i databasen.
Andet	Brugeroplysninger består af MA-nummer og en adgangskode.

BRUGER

USE CASE: KURSUSTILMELDING

Use case	**Kursustilmelding**
Aktivering	Brugeren klikker på “Tilmeld”-knappen.
Prækondition	Brugeren er autentificeret og opfylder kravene for at tilmelde sig kurset
Primær sti	1. Brugeren benytter søgefunktionen til at finde et specifikt kursus. 2. Serveren returnerer kurset på baggrund af søgningen. 3. Brugeren klikker på “Tilmeld” ud for det ønskede uddannelsessted. 4. Serveren verificerer at brugeren opfylder kravene for at tilmelde sig kurset, samt at kurset er tilgængeligt. 5. Opfylder brugeren kravene og kurset er tilgængeligt, vil brugeren få en bekræftelse på tilmeldingen.
Alternative stier	4. Opfylder brugeren ikke kravene eller er kurset ikke tilgængeligt, vil dette blive vist i form af en fejlmeddelelse i skridt 2.
Postkondition	Brugeren er tilmeldt kurset
Særtilfælde	Hvis brugeren ikke trykker på tilmeld, vil formulardataene blive glemt, når brugeren klikker videre på en ny side. Der vil ikke ske ændringer i databasen.
Andet	Et kursus er tilgængeligt hvis det har ledige pladser, ikke er aflyst og endnu ikke er startet.

USE CASE: GENNEMFØRTE KURSER

Use case	**Gennemførte kurser**
Aktivering	Brugeren klikker på linket “Gennemførte kurser”.
Prækondition	Brugeren er autentificeret
Primær sti	1. Brugeren sender en forespørgsel til serveren omkring gennemførte kurser

	2. På basis af MA-nummeret, laves et opslag i databasen der returnerer alle gennemførte kurser 3. Resultatet præsenteres for brugeren som en liste
Alternative stier	Ingen
Postkondition	Brugeren kan se en liste over gennemførte kurser
Særtilfælde	Har brugeren ikke gennemført nogen kurser, vil listen være tom
Andet	Intet

USE CASE: TILGÆNGELIGE KURSER

Use case	**Tilgængelige kurser**
Aktivering	Brugeren klikker på linket "Tilgængelige kurser"
Prækondition	Brugeren er autentificeret
Primær sti	1. Brugeren sender en forespørgsel til serveren omkring gennemførte kurser. 2. Systemet laver et opslag efter alle kurser som brugeren har gennemført, baseret på brugerens MA-nummer. Dette sæt kaldes "gennemførte" . Systemet laver herefter et opslag som returnerer fællesmængden af kurser som ikke har nogen forudsatte kurser, samt de kurser som er forudsat af "gennemførte". Dette sæt kaldes "mulige". Der returneres herefter en liste af kurser som er mængden "mulige" fratrukket mængden "gennemførte". 3. Resultatet præsenteres for brugeren som en liste.
Alternative stier	Ingen
Postkondition	Brugeren kan se en liste over tilgængelige kurser.
Særtilfælde	Har brugeren ikke opfyldt kravene til nogen kurser, vil listen være tom.
Andet	I den returnerede liste vil kurser som brugeren allerede har gennemført være filtreret fra.

USE CASE: MANGLENDE KURSER

Use case	**Manglende kurser**
Aktivering	Brugeren klikker på linket "Manglende kurser"
Prækondition	Brugeren er autentificeret
Primær sti	1. Brugeren sender en forespørgsel til serveren omkring manglende kurser. 2. På basis af MA-nummeret, laves et opslag i databasen der returnerer alle kurser, der endnu ikke er gennemført og som er krævede for at tilmelde sig det pågældende kursus. 3. Resultatet præsenteres for brugeren som en liste og samtidig vises et afkrydsningsfelt, hvor brugeren kan til- eller fravælge at inkludere kurser der allerede er bestået.
Alternative stier	2. I stedet for at vise krævede kurser for et andet kursus, gør denne use case sig også gældende ved krævede kurser for en funktion.
Postkondition	Brugeren kan se en liste over manglende kurser

Særtilfælde	Mangler brugeren ingen kurser, vil listen være tom
Andet	Uddannelsesdirektivet for et kursus kan i sig selv indeholde andre uddannelsdirektiver med dertilhørende kursus. Heraf kan et kursus også indeholde andre kurser.

USE CASE: REDIGER PERSONOPLYSNINGER

Use case	**Rediger personoplysninger**
Aktivering	Brugeren klikker på linket "Rediger personoplysninger"
Prækondition	Brugeren er autentificeret.
Primær sti	1. Brugeren har nu mulighed for at rette i felterne og trykke "Gem" 2. Serveren verificerer formulardata for ugyldigt input 3. Er alle felter gyldige, vil personoplysningerne i blive opdateret. 4. Brugeren viderestilles til oversigten over de gemte personoplysninger.
Alternative stier	3. Er et eller flere felter ugyldige, vil brugeren få vist en fejlmeddelelse om dette.
Postkondition	Personoplysningerne er opdateret og databasen indeholder nu kun de nyeste oplysninger
Særtilfælde	Nogle oplysninger såsom MA-nummer og fødselsdag ændres aldrig og derfor har brugeren heller ikke mulighed for at rette disse.
Andet	Brugeren kan til enhver tid afbryde og dermed undgå ændringer til databasen.

USE CASE: STATUS FOR TILMELDTE KURSER

Use case	**Status for tilmeldte kurser**
Aktivering	Brugeren klikker på linket "Status for tilmeldte kurser"
Prækondition	Brugeren er autentificeret.
Primær sti	1. En forespørgsel om tilmeldte kurser sendes til serveren på baggrund af brugerens MA-nummer 2. Serveren returnerer en liste over alle kurser brugeren har tilmeldt sig. 3. Listen præsenteres for brugeren som en liste.
Alternative stier	Ingen
Postkondition	Brugeren kan se en liste over tilmeldte kurser samt deres status
Særtilfælde	Har brugeren ikke tilmeldt sig nogen kurser, vil listen være tom
Andet	Intet

AFDELINGSLEDER

USE CASE: TILMELD FLERE TIL KURSUS

Use case	**Tilmeld flere til kursus**
Aktivering	Afdelingslederen klikker på "Tilmeld flere"-knappen.
Prækondition	Afdelingslederen er autentificeret og opfylder dermed kravene til at tilmelde andre til kurser.
Primær sti	1. Afdelingslederen benytter søgefunktionen til at finde et specifikt kursus. 2. Serveren returnerer kurset på baggrund af søgningen. 3. Afdelingslederen klikker på "Tilmeld" ud for det ønskede uddannelsessted. 4. Serveren returnerer en liste over brugere i afdelingen, som endnu ikke har gennemført kurset eller allerede er tilmeldt. 5. Afdelingslederen vælger en eller flere brugere på listen og trykker "Tilmeld". 6. Systemet sender en bekræftelse til de tilføjede brugeres email.
Alternative stier	Ingen
Postkondition	Brugerne er tilmeldt kurset og har fået tilsendt bekræftelse på dette.
Særtilfælde	Hvis afdelingslederen ikke trykker på tilmeld, vil formulardataene blive glemt, når afdelingslederen klikker videre på en ny side. Der vil ikke ske ændringer i databasen.
Andet	

USE CASE: GENNEMFØRTE KURSER FOR AFDELING

Use case	**Gennemførte kurser for afdeling**
Aktivering	Afdelingslederen klikker på linket "Gennemførte kurser"
Prækondition	Afdelingslederen er autentificeret og har dermed ret til at se andres gennemførte kurser
Primær sti	1. Afdelingslederen sender en forespørgsel til serveren omkring gennemførte kurser for brugere i vedkommendes afdeling. 2. Serveren danner en liste over brugere og returnerer alle gennemførte kurser for disse brugere. 3. Resultatet præsenteres for afdelingslederen som en matrix, hvor brugere udgør den ene akse, mens kurser udgør den anden.
Alternative stier	Ingen
Postkondition	Afdelingslederen kan se en liste over afdelingens gennemførte kurser
Særtilfælde	Ingen
Andet	Intet

KURSUSANSVARLIG

USE CASE: OPDATER KURSUS

Use case	**Opdater kursus**
Aktivering	Kursusansvarlig klikker på linket "Opdater kursus"
Prækondition	Kursusansvarlig er autentificeret og har tilstrækkelige rettigheder til at opdatere kurser i databasen
Primær sti	1. Serveren returnerer kurset fra databasen og udfylder en formular med informationerne fra kurset, som vises til kursusansvarlig. 2. Kursusansvarlig har nu mulighed for at rette i felterne og trykke "Gem" 3. Serveren verificerer formulardata for ugyldigt input 4. Er alle felter gyldige, vil kurset blive opdateret og ved søgning vil den nye version blive vist 5. Kursusansvarlig viderestilles til oversigten over det pågældende kursus
Alternative stier	3. Er et eller flere felter ugyldige, vil brugeren få vist en fejlmeddelelse om dette.
Postkondition	Kurset er opdateret og den opdaterede version kan nu findes ved søgning
Særtilfælde	Ingen
Andet	Kursusansvarlig kan til enhver tid afbryde og dermed undgå ændringer i databasen

USE CASE: OPRET KURSUSINSTANS

Use case	**Opret kursusinstans**
Aktivering	Kursusansvarlig klikker på linket "Opret kursusinstans
Prækondition	Kursusansvarlig er autentificeret og har tilstrækkelige rettigheder til at oprette kursusinstanser i databasen.
Primær sti	1. Serveren returnerer en liste over kurser som kursusansvarlig kan oprette kursusinstanser af. 2. Kursusansvarlig udfylder en formular hvor kursus, tid og sted angives og trykker "Opret". 3. Serveren tjekker om en kursusinstans allerede eksisterer med den indtastede information eller om et felt er ugyldigt. 4. Eksisterer det ikke, vil kursusinstansen nu oprettes.
Alternative stier	4. Eksisterer det allerede eller er et eller flere felter ugyldige, vil brugeren få vist en fejlmeddelelse om dette og gå tilbage til skridt 2.
Postkondition	Kursusinstansen er oprettet og er nu åben for tilmelding
Særtilfælde	Afbrydes der før skridt 4, vil databasen ikke være ændret.
Andet	Intet

USE CASE: OPDATER KURSUSINSTANS

Use case	Opdater kursusinstans
Aktivering	Kursusansvarlig klikker på linket "Opdater kursusinstans"
Prækondition	Kursusansvarlig er autentificeret og har tilstrækkelige rettigheder til at opdatere kursusinstanser oprettet af vedkommende selv.
Primær sti	1. Serveren returnerer kursusinstansen fra databasen og viser en formular udfyldt med informationerne om instansen. 2. Kursusansvarlig har mulighed for at rette i felterne og trykke "Gem". 3. Serveren verificerer formulardata for ugyldigt input. 4. Er alle felter gyldige, vil kursusinstansen blive opdateret og ved søgning vil den nye version blive vist 5. Kursusansvarlig viderestilles til oversigten over det pågældende kursus
Alternative stier	4. Er et eller flere felter ugyldige, vil brugeren få vist en fejlmeddelelse om dette.
Postkondition	Kursusinstansen er opdateret og den opdaterede version kan nu findes ved søgning.
Særtilfælde	Ingen
Andet	Kursusansvarlig kan til enhver tid afbryde og dermed undgå ændringer i databasen.

USE CASE: REGISTRER GENNEMFØRT ELLER DUMPET KURSIST

Use case	Registrer gennemført eller dumpet kursist
Aktivering	Kursusansvarlig klikker på linket "Registrer kursister"
Prækondition	Kursusansvarlig er autentificeret og har tilstrækkelige rettigheder til at registrere kursister på egne kursusinstanser.
Primær sti	1. Serveren returnerer en liste over alle kursusinstanser den kursusansvarlige er ansvarlig for. 2. Kursusansvarlig vælger en kursusinstans. 3. Serveren returnerer alle kursister på kursusinstansen som en liste. 4. Listen præsenteres for kursusansvarlig. 5. Kursusansvarlig har nu mulighed for at krydse af i felterne og trykke "Gem". 6. Serveren opdaterer status for gennemførsel. 7. Kursusansvarlig viderestilles til en side med status for ændringen.
Alternative stier	Ingen
Postkondition	Kurset er opdateret og den opdaterede version kan nu findes ved søgning.
Særtilfælde	Afbrydes der før skridt 5, vil ingen ændringer blive gemt til databasen.
Andet	Kursusansvarlig kan til enhver tid afbryde og dermed undgå ændringer til databasen.

KURSUSADMINISTRATOR

USE CASE: OPRET KURSUS

Use case	**Opret kursus**
Aktivering	Brugeren klikker på linket "Opret kursus"
Prækondition	Brugeren er autentificeret og har tilstrækkelige rettigheder til at oprette kurser i databasen
Primær sti	1. Brugeren udfylder en formular indeholdende kort og lang titel, som sendes til serveren. 2. Serveren tjekker om et kursus allerede eksisterer med den indtastede information. 3. Eksisterer det ikke, vil kurset nu oprettes og brugeren gives muligheden for at vedhæfte et UDIR. 4. Brugeren udfylder formularen for UDIR og sender den til serveren 5. Serveren verificerer det indtastede for ugyldige felter. 6. Hvis alle felter er gyldige vil kurset med dertilhørende UDIR nu være tilføjet til databasen
Alternative stier	3. Eksisterer det allerede eller er et eller flere felter ugyldige, vil brugeren få vist en fejlmeddelelse om dette og gå tilbage til skridt 1. 6. Hvis et felt er ugyldigt, vil en fejlmeddelelse blive vist og brugeren går tilbage til skridt 4.
Postkondition	Kurset er oprettet og kan nu findes ved søgning, med eller uden vedhæftet UDIR
Særtilfælde	Afbrydes der før skridt 3, vil databasen ikke være ændret, men afbrydes der mellem 3 og 5 vil kurset være oprettet, dog uden UDIR.
Andet	Intet

USE CASE: OPRET UDDANNELSESDIREKTIV

Use case	**Opret uddannelsesdirektiv**
Aktivering	Brugeren klikker på linket "Opret uddannelsesdirektiv"
Prækondition	Brugeren er autentificeret og har tilstrækkelige rettigheder til at oprette uddannelsesdirektiver i databasen
Primær sti	1. Brugeren udfylder en formular der beskriver et uddannelsesdirektiv. 2. Formularen valideres hos brugeren, for at sikre at alle nødvendige felter er udfyldt inden den sendes til serveren. 3. Den validerede formular sendes til serveren. 4. Serveren tjekker om et uddannelsesdirektiv med samme nummer allerede eksisterer. 5. Uddannelsesdirektivet oprettes i databasen og brugeren gives muligheden for at tilføje det til et kursus gennem use casen "Tilføj kursus til uddannelsesdirektiv"
Alternative stier	3. Hvis et krævet felt ikke er udfyldt eller er udfyldt forkert, vil brugeren få vist en fejlmeddelelse om dette og blive anmodet om at rette fejlen i skridt 1. 5. Eksisterer der allerede et andet uddannelsesdirektiv med samme UDIR-nummer, vil brugeren få vist en fejlmeddelelse om dette og blive anmodet om at rette fejlen i skridt 1.
Postkondition	Uddannelsesdirektivet er oprettet og kan nu findes ved søgning, med eller uden vedhæftet kursus

Særtilfælde	Brugeren kan på et hvilket som helst tidspunkt afbryde oprettelsen af uddannelsesdirektivet og dermed ændres databasen ikke.
Andet	Et uddannelsesdirektiv er beskrevet ved et UDIR-nummer, formål, mål, indhold, samt en række ekstra punkter der ikke allesammen er obligatoriske.

USE CASE: OPDATER UDDANNELSESDIREKTIV

Use case	**Opdater uddannelsesdirektiv**
Aktivering	Brugeren klikker på linket "Opdater uddannelsesdirektiv"
Prækondition	Brugeren er autentificeret og har tilstrækkelige rettigheder til at opdatere uddannelsesdirektiver i databasen
Primær sti	1. Brugeren finder uddannelsesdirektivet ved søgning. 2. Serveren returnerer en formular udfyldt med informationer om uddannelsesdirektivet, som vises til brugeren. 3. Brugeren har nu mulighed for at rette i felterne og trykke "Gem". 4. Formularen valideres hos brugeren, for at sikre at alle nødvendige felter er udfyldt inden den sendes til serveren. 5. Er alle felter gyldige, skal brugeren tage stilling til, om ændringerne er så markante at det giver grundlag for at oprette et nyt uddannelsesdirektiv i stedet for at opdatere det gamle. 6. Er dette ikke tilfældet, bliver det gamle blot opdateret.
Alternative stier	5. Er et eller flere felter ugyldige, vil brugeren få vist en fejlmeddelelse om dette og gå tilbage til skridt 3. 6. Hvis et nyt uddannelsesdirektiv skal oprettes, gemmes det i databasen og alle kurser der henviser til det gamle uddannelsesdirektiv ændres så de i stedet henviser til det nye.
Postkondition	Kurset er opdateret og den opdaterede version kan nu findes ved søgning,
Særtilfælde	Brugeren kan på et hvilket som helst tidspunkt afbryde opdateringen af uddannelsesdirektivet og dermed ændres databasen ikke.
Andet	Et uddannelsesdirektiv er beskrevet ved et UDIR-nummer, formål, mål, indhold, samt en række ekstra punkter der ikke allesammen er obligatoriske.

USE CASE: UDPEG KURSUSANSVARLIG

Use case	**Udpeg kursusansvarlig**
Aktivering	Brugeren klikker på linket "Udpeg kursusansvarlig" på en kursusside
Prækondition	Kursusadministratoren er autentificeret og har dermed tilstrækkelige rettigheder til at udpege kursusansvarlige
Primær sti	1. Kursusadministratoren finder kurset ved søgning. 2. Systemet returnerer kursussiden fra databasen. 3. Kursusadministratoren klikker på "Udpeg kursusansvarlig". 4. Systemet returnerer en liste over potentielle kursusansvarlige. 5. Kursusadministratoren vælger en eller flere brugere på listen og trykker "Gem". 6. Systemet tilføjer de valgte brugere til listen over kursusansvarlige på det pågældende

	kursus, og giver dem dermed ret til at oprette kursusinstanser og ændre i disse. 7. Systemet viderestiller kursusadministratoren til kursussiden.
Alternative stier	6. Er ingen brugere valgt, vil en fejlmeddelse vise dette og give mulighed for at vælge igen.
Postkondition	Kurset er opdateret og den opdaterede version kan nu findes ved søgning,
Særtilfælde	Brugeren kan på et hvilket som helst tidspunkt afbryde opdateringen af uddannelsesdirektivet og dermed ændres databasen ikke.
Andet	Et uddannelsesdirektiv er beskrevet ved et UDIR-nummer, formål, mål, indhold, samt en række ekstra punkter der ikke allesammen er obligatoriske. Markante ændringer defineres ud fra en vurdering fra gang til gang.

BRUGERADMINISTRATOR

USE CASE: OPRET BRUGER

Use case	**Opret bruger**
Aktivering	Brugeradministratoren klikker på linket “Opret bruger”
Prækondition	Brugeradministratoren er autentificeret og har dermed tilstrækkelige rettigheder til at oprette brugere i databasen
Primær sti	1. Brugeradministratoren udfylder en formular med personoplysninger om brugeren og trykker “Opret”. 2. Serveren tjekker om en bruger allerede eksisterer med den indtastede information eller om et eller flere felter er ugyldige. 3. Eksisterer den ikke og er alle felter gyldige, oprettes brugeren med en autogenereret adgangskode, der sendes til brugerens email. 4. Brugeradministratoren gives muligheden for at uddele roller til brugeren. 5. Systemet returnerer en liste over tilgængelige roller og præsenterer dem for brugeradministratoren. 6. Brugeradministratoren vælger eventuelle roller og klikker “Gem”.
Alternative stier	3. Eksisterer brugeren allerede eller er et eller flere felter ugyldige, vil brugeradministratoren få vist en fejlmeddelelse om dette og gå tilbage til skridt 1.
Postkondition	Brugeren er oprettet og kan nu logge ind med med brugerens MA-nummer og adgangskoden sendt til brugerens email.
Særtilfælde	Afbrydes der i skridt 1, vil databasen ikke være ændret, men afbrydes der mellem skridt 3 og 5 vil brugeren være oprettet, dog uden nogen roller.
Andet	Intet

ADGANG TIL SYSTEMET

Systemet skal designes på en måde så brugere kan få adgang gennem NemID. CCMS skal med andre ord ikke have sit eget login-system, hvor brugeren skal have en specifik adgangskode til netop dette system, men han skal i stedet kunne benytte en autentificeringsmetode som baserer sig på den digitale signatur som personen allerede har til rådighed.

DETALJEREDE IKKE-FUNKTIONELLE KRAV

Nedenfor gennemgåes kerneentiteterne for CCMS. Bemærk at selvforklarende attributter på entiteterne ikke er beskrevet i detaljer.

BRUGER

Indhold	Type	Beskrivelse	Kommentar
MA-nummer	Tal	Medarbejdernummer	
CPR-nummer	Tal	Personnummer	
Fornavn	Tekst		
Efternavn	Tekst		
Adresse	Tekst		
Postnummer	Tal		
Telefonnummer	Tal		
Mobilnummer	Tal		
Email	Tekst		
Fødselsdag	Dato		
Medarbejdergruppe	Tekst	Fx “frivillig” eller “ansat”	
Grad	Tekst	Medarbejderens rang/grad	
Organisation	Tekst	Tilknytning i organisationen	
Primær funktion	Funktion (Entitet)	Stillingsbetegnelse/ arbejdsfunktion i organisationen	
Sekundær funktion	Funktion (Entitet)	Stillingsbetegnelse/ arbejdsfunktion i organisationen	
Tertiær funktion	Funktion (Entitet)	Stillingsbetegnelse/ arbejdsfunktion i organisationen	
Køretøjstype	Tekst	Fx “bil” eller “motorcykel”	Kan optræde flere gange

Køretøjs-registreringsnummer	Tekst		Kan optræde flere gange
Bestået Q-nummer			Kan optræde flere gange

KURSUS

Indhold	Type	Beskrivelse	Kommentar
Langt navn	Tekst	Kursets lange navn	
Kort navn	Tekst	Kursets korte navn	
Nuværende uddannelsesdirektiv	Udir (Entitet)		
Tidligere uddannelsesdirektiv	Udir (Entitet)		Kan optræde flere gange
Fagplan	Fagplan (Entitet)		Kan optræde flere gange
Dokument	Dokument (Entitet)		Kan optræde flere gange
Kursusafholdelse	Kursus-afholdelse (Entitet)		Kan optræde flere gange
Kursusansvarlig	Bruger (Entitet)		Kan optræde flere gange

FUNKTION

Indhold	Type	Beskrivelse	Kommentar
Navn	Tekst	Funktionens navn	
Krævet kursus	Kursus (Entitet)	Forudsat kursus for at en medarbejder kan bestride en funktion	Kan optræde flere gange

UDIR

Indhold	Type	Beskrivelse	Kommentar
Tekstblok	Tekst	Grupperet tekstblok med indhold om uddannelsesdirektivet	Kan optræde flere gange

Q	Tal	Q-nummer som uddannelsen giver	Kan optræde flere gange
Udir-nummer	Tal	Uddannelsesdirektiv-nummer	

DOKUMENT

Indhold	Type	Beskrivelse	Kommentar
Tekstblok	Tekst	Dokumentets indhold	

FAGPLAN

Indhold	Type	Beskrivelse	Kommentar
Tekstblok	Tekst	Dokumentets indhold	
Dato	Dato	Datoen hvor dokumentet blev oprettet	
Nummer	Tal	Fagplanens nummer	

KURSUSAFHOLDELSE

Indhold	Type	Beskrivelse	Kommentar
Kursus	Kursus (Entitet)	Det kursus som afholdes	
Dato	Dato	Den dato som kurset starter	
Skole	Tekst	Den skole om afholder kurset	
Kursusansvarlig	Bruger (Entitet)	Den bruger som afholder dette kursus	
Kursusstatus	Tekst	Om kurset er aflyst, fyldt og tilsvarende.	

SIKKERHED

Kommunikationen mellem klient og server vil have et lag af autentificering, for at forebygge uatoriserede skrive/slette-operationer til databasen. For at skrive eller slette fra databasen, kræves der derfor et tilstrækkeligt rettighedsniveau. Læsning fra databasen er som udgangspunkt uden begrænsninger. En enkelt undtagelse findes i form af personlige oplysninger om en bruger, der kun er tilgængelige for brugeren selv, samt administratorer, for hvem det eventuelt måtte være hensigtsmæssigt at have denne adgang.

For at eliminere muligheden for at manipulere med data før de skrives til databasen, vil kun den meste basale input-validering ske hos brugeren, hvilket omfatter ting som korrekt formatering af datoer og at obligatoriske felter ikke er tomme. Dette gøres af hensyn til ydelsen af systemet, da det er hurtigere at validere disse ting hos klienten, samtidig med at det ikke belaster serveren unødvendigt. Mere kritiske ting som at sikre at den pågældende bruger har rettigheder til at foretage en given operation på databasen, vil ske på baggrund af validering på serveren.

ANALYSE

Med udgangspunkt i den kravspecifikation som vi har formuleret, vil vi i dette kapitel analysere forskellige af de delproblemer som kravspecifikationen har afdækket. Derudover vil vi designe en datamodel som kan indeholde den information der er behov for, i henhold til at kunne implementere en løsning som er brugbar for hjemmeværnet.

RESEARCH

Hjemmeværnet har på nuværende tidspunkt ikke en central IT-løsning til at håndtere uddannelsen af dets medlemmer. I stedet benyttes en række forskellige teknologier som skal arbejde sammen for at opnå en komplet løsning. I dette afsnit vil de enkelte dele blive beskrevet kort, hvordan vi har brugt dem, samt hvorfor de ikke udgør en optimal løsning.

Overordnet set benytter Hjemmeværnet:

- Hvs-info.dk
- Access-frontend
- Excel-database

HVS-INFO.DK - WWW

På denne side findes de fleste informationer om hjemmeværnets kurser. Her er det muligt at finde fagplaner og uddannelsesdirektiver for kurser, der beskriver detaljer som læringsmål og indhold af et kursus, samt hvor lang tid et kursus tager, om der gives løn mm. Det er ligeledes muligt at se uddannelsesvejen og adgangskravene for at tage et bestemt kursus eller påtage sig en specifik funktion i Hjemmeværnet i form af hvilke kurser der skal være opfyldt.

Der er taget udgangspunkt i denne hjemmesides funktioner i forhold til de gængse arbejdsgange i Hjemmeværnet, for at sikre at systemet dækker disse. En funktion kunne være muligheden for at se tid og sted for næste afholdelse af et kursus, samt adgangskravene til dette.

HJEMMEVÆRNETS UDDANNELSESDATABASE - ACCESS

Hjemmeværnets uddannelsesdatabase er navnet på et grafisk interface udarbejdet i Microsoft Access. Dette værktøj indeholder i sig selv intet data, men bruges udelukkende til at præsentere data på en mere overskuelig måde. Det er i stedet afhængigt af ekstern data, der skal importeres separat. Det indeholder en masse logik til forespørgsler, hvor det grafiske interface lægger sig som et lag over logikken. Dette interface sikrer at medarbejdere kan benytte databasen, selv hvis de ikke er erfarne it-brugere.

Værktøjets formål er at give en medarbejder, som ofte er en afdelingsleder eller anden med ansvar for afdelingens medlemmer, mulighed for at danne sig et overblik.

Når data skal indlæses, åbnes Excel-filen, hvorefter man bruger copy/paste til at flytte alt data over i Access-filen. Når data er indlæst i Access-filen, kan filen gemmes med det aktuelle data, så det ikke skal importeres hver gang filen åbnes.

Access-filen kan hentes uden data på Hjemmeværnets hjemmeside hjv.dk, når man er logget ind. På denne side nævnes også at Access-filen uden data frit må sendes videre, mens den under ingen omstændigheder må distribueres, når data er importeret.

Access-filen giver, takket være dets interface, et ret godt overblik over de centrale features der bør være i systemet, udover dem der dækkes af hvs-info.

DATA TIL HJEMMEVÆRNETS UDDANNELSESDATABASE - EXCEL

Hjemmeværnets Uddannelsesdatabase udarbejdet i Access, indeholder som nævnt intet data i sig selv. I stedet benyttes Excel til at holde styr på hvilke kurser de enkelte medlemmer har opnået Q i.

Data opbevares som en flad liste i et regneark, hvor hver eneste linje indeholder den komplette information om en person. Kun enkelte kolonner ændres for hver linje, såsom Q-nummer og kursusnavn.

Regnearket med data er opdelt på underafdelings niveau, så data frit kan distribueres blandt medlemmer i en underafdeling, uden at man på denne måde opnår indsigt i andre underafdelinger. At databasen er splittet i flere hundrede filer pga. de mange underafdelinger, skyldes at den udelukkende bliver distribueret via email. At sende hele databasen som en enkelt fil, ville hurtigt blive et problem, da størrelsen på denne ville overstige grænserne for vedhæftninger til email hos de fleste udbydere.

Denne fil kan ikke downloades nogen steder og skal i stedet sendes som email, men på hjv.dk, på samme underside som Access-filen hentes, opfordres der til varsom distribuering af denne fil over email.

Bliver en ny version af Excel-filen tilgængelig, skal denne videresendes til samtlige medarbejdere der måtte have interesse i at have den, da de ellers sidder med gammelt data.

NUVÆRENDE ARBEJDSGANGE

Alle medlemmer af Hjemmeværnet må som udgangspunkt være i besiddelse af uddannelsesdatabasen, udfyldt med data fra underafdelingen de selv er medlem af. Den enkelte medarbejder bruger dette værktøj til at danne sig et overblik over hvilke kurser vedkommende mangler. For at få en grundigere forklaring af et kursus, kan der herefter søges på hvs-info.dk. På hvs-info findes uddannelsesdirektivet der beskriver kurset i detaljer, ligesom det også er muligt at se planlagte kursussteder og datoer for det specifikke kursus.

MEDLEMMER TILMELDER SIG KURSUS

Vil medlemmet gerne tilmeldes kurset, kontaktes den overordnede i underafdelingen, som videregiver tilmeldingen til distriktet. Der kan også tilmeldes direkte til distriktet, men dette anbefales ikke.

AFDELINGSLEDEREN TILMELDER MEDLEMMER TIL KURSUS

For afdelingslederen ville en typisk arbejdsopgave være at se, hvor mange af afdelingens medlemmer der mangler et givent kursus og herefter tilmelde dem til dette. Oversigten kunne også være mere generel og vise samtlige manglende kurser frem for ét specifikt kursus. Fælles for disse er at afdelingslederen ikke direkte kan tilmelde medlemmer uden deres accept og skal derfor først indhente denne, for herefter at indberette tilmeldingen til distriktet. Dette er grunden til at det ikke anbefales medlemmer at tilmelde sig kurser udenom afdelingslederen, da afdelingslederen kan være i gang med at indhente accept fra afdelingens medlemmer med henblik på at disse udgør et samlet hold på et kursus.

UDDANNELSESDATABASEN OPDATERES

Databasen holdes ajourført på distriktsniveau, hvor medlemmer fra alle underafdelinger bliver opdateret, når de består et kursus og opnår Q. Herefter kan Excel-filen som bruges til Access-uddannelsesdatabasen eksporteres, som efter hver opdatering bør distribueres til alle der berøres af ændringer.

Har et medlem i en afdeling lige bestået et obligatorisk kursus, mens afdelingslederen gennemgår listen, vil kurset ikke være anført som bestået for dette medlem, indtil afdelingslederen får tilsendt den nye og opdaterede Excel-fil.

VORES LØSNING - KORT FORTALT

Den nuværende løsning baserer sig på decentralisering af data, som spredes uden nogen kontrol med hvem der har adgang til data. I samme ombæring er der en stor risiko for inkonsistent data, idet flere personer kan have fået den samme version tilsendt, men derefter lavet ændringer i databasen. Herefter sidder de enkelte medarbejdere med hver sin version. Desuden er løsningen ineffektiv og unødigt tidskrævende.

I stedet for at benytte en decentraliseret løsning, hvor databasens egentlige indhold skal distribueres over mail, samles det i en enkelt database, der sikrer at man altid opererer med de aktuelle data, samt ikke er afhængig af at skulle have den nyeste version tilsendt.

FREMGANGSMÅDE

I forbindelse med vores gennemgang af Hjemmeværnets systemer har vi overordnet set benyttet fire kilder til information:

1. hvs-info.dk
2. Access-database
3. hjv.dk
4. Tilknyttede medarbejdere ved hjemmeværnet

HVS-INFO.DK

Dette website indeholder information om alle kurser i Hjemmeværnet. Vi har benyttet websitet til få information om hvordan uddannelsessystemet hænger sammen, og hvordan relationerne er mellem uddannelser. I dette analysekapitel og i andre dele af rapporten henviser vi til konkrete kurser og sammenhænge som kan findes på dette website. De fleste informationer findes hvis man vælger “Uddannelser” efterfulgt af “Alle uddannelser”. En del information om relationerne mellem uddannelser kan også findes hvis man vælger “Uddannelsesstruktur” efterfulgt af “Uddannelsesveje”.

ACCESS-DATABASE

Som omtalt tidligere benyttes en Access-database til at registrere information om hvilke medarbejdere der har gennemført hvilke kurser. Denne database har et grafisk-interface og man kan navigere og finde konkret information om gennemførte og manglende kurser for konkrete medarbejde. Denne database er ikke offentligt tilgængelig, men vi har i Appendix indsat et skærmbillede af forsiden af denne database.

HJV.DK

Denne side skrives nyheder for hjemmeværnsansatte og medarbejdere, og det indeholder desuden et stort antal dokumenter med relevans for Hjemmeværnets ansatte. Der er dels offentlig information, og information som kræver at man er logget ind.

Vi har bl.a. benyttet dette website til at finde ud af hvilke informationer som Hjemmeværnet registrerer om sine medarbejdere. I appendix findes et billede fra brugersiden på det lukkede system.

MEDARBEJDERE VED HJEMMEVÆRNET

Vi har primært benyttet vores kontakter indenfor Hjemmeværnet til at få information om hvilke funktioner de har brug for i et nyt system - Disse informationer ligger til grund for kravspecifikationen. Desuden har vi med hjælp fra disse personer fået afklaret tvivlspørgsmål omkring deres nuværende systemer og hvordan de benytter dem.

FEJL OG INKONSISTENSER I DE NUVÆRENDE LØSNINGER

I vores gennemgang og analyse af de nuværende systemer har vi haft brug for at undersøge konkrete eksempler på uddannelser og deres relationer. Vi har benyttet hvs-info.dk til at finde denne information, og ved den lejlighed har vi også fundet et stort antal inkonsistenser og fejl som indikerer at den nuværende kursus-hjemmeside er vedligeholdt manuelt, og at det ikke er systemet som relaterer uddannelserne til hinanden. I appendix findes et lille udvalg af eksempler på inkonsistenser.

Det store antal inkonsistenser har vanskeliggjort vores forsøg på at finde system i uddannelsernes opbygning, med henblik på at bygge en datamodel som kan understøtte uddannelserne. Samtidig har de understreget behovet for det system vi specificerer.

KURSUSHÅNDTERING

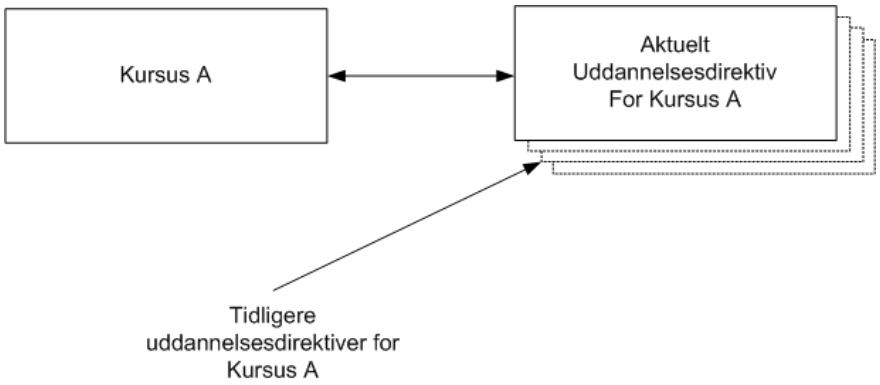
Vi vil som udgangspunkt beholde den repræsentation for kurser som i øjeblikket benyttes af Hjemmeværnet, og som kan ses på http://hvs-info.dk/. Denne har en række uhensigtsmæssigheder som formentlig bunder i at kurser er indsat og håndteret manuelt, uden at man har haft en tilstrækkeligt klar instruktion omkring hvordan informationerne omkring et kursus skal struktureres, og uden at man har haft et computersystem som har tvunget brugerne til at indordne sig under systemets datastruktur. Vi vil i senere afsnit komme nærmere ind på disse uhensigtsmæssigheder, og hvordan de giver udfordringer for os.

Det forholder sig sådan, at på trods at de uhensigtsmæssigheder som vi har opdaget, herunder også hvordan datamodellen bliver meget mere kompliceret og vanskelig at gennemskue, så ønsker man fra Hjemmeværnets side at vores system skal indeholde kurser på den form som de findes i deres nuværende system. I den forbindelse er det nødvendigt at vi opfatter et kursus som noget stabilt, der kan eksistere i mange år, men samtidig skal det være muligt at ændre i en uddannelses formelle indhold, og det skal være muligt at finde indholdet for tidligere versioner af et kursus. Modellen for kurser inddeler konceptet "Et kursus" i to separate entiteter:

- kursus
- udir (Uddannelsesdirektiv)

Kursus-entiteten er den stabile del af et kursus som kan eksistere i mange år, og som kun indeholder helt overordnede informationer, så som kursets titel. Uddannelsesdirektivet indeholder alle øvrige detaljer om kurset.

En illustration at dette forhold ses nedenfor:



FIGUR 2 - KURSUS-ENTITETEN ER DEN STABILE DEL AF ET KURSUS. UDDANNELSESDIREKTIVET INDEHOLDER ALLE ØVRIGE DETALJER OM KURSET

Der findes to konceptuelle relationer mellem uddannelser, nemlig følgende:

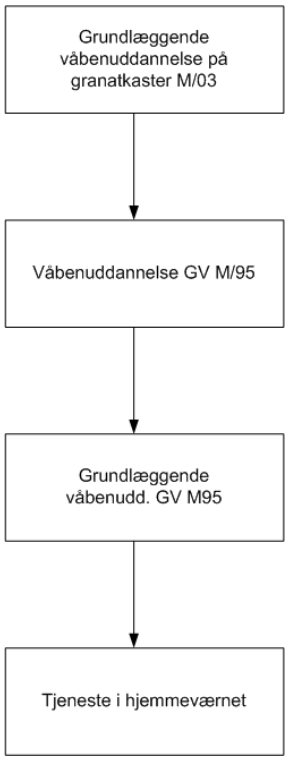
1. Kræver (forudsætter)
2. Indeholder

KURSUSFORUDSÆTNINGER

Det er almindeligt at et kursus stiller krav om at medarbejderen har gennemført et eller flere andre kurser, inden vedkommende påbegynder det specifikke kursus. Vi har for simplicitetens skyld valgt kun at registrere de kurser som er øverst i uddannelseshierarkiet af krævede uddanelser for en bestemt uddannelse.

I eksemplet til højre ville man altså for kurset "Grundlæggende våbenuddannelse på granatkaster M/03" kun registerere "Våbenuddannelse GV M/95". Når en medarbejder ønsker at vide hvilken serie af uddannelser der ligger foran vedkommende inden kurset "Grundlæggende våbenuddannelse på granatkaster M/03" kan tages, vil man med andre ord lade systemet rekursivt navigere ned i uddannelseshierarkiet, og finde ud af hvilke kurser medarbejderen mangler.

Med den model bliver uddannelseshierarkiet væsentligt nemmere at vedligeholde.



FIGUR 3 - FORUDSÆTNINGER

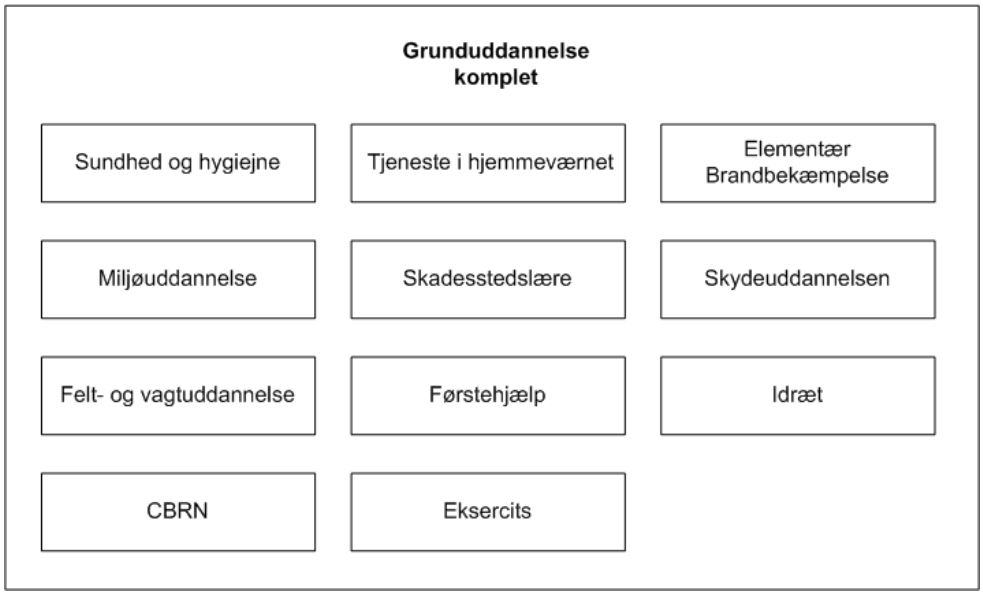
KURSER MED FÆLLES INDHOLD

Der findes kurser som har indhold til fælles. Dette kan foregå på tre måder:

1. Et kursus indeholder et eller flere kurser som også kan tages selvstændigt
2. Flere kurser deles om fagplaner
3. Flere kurser deles om dokumenter

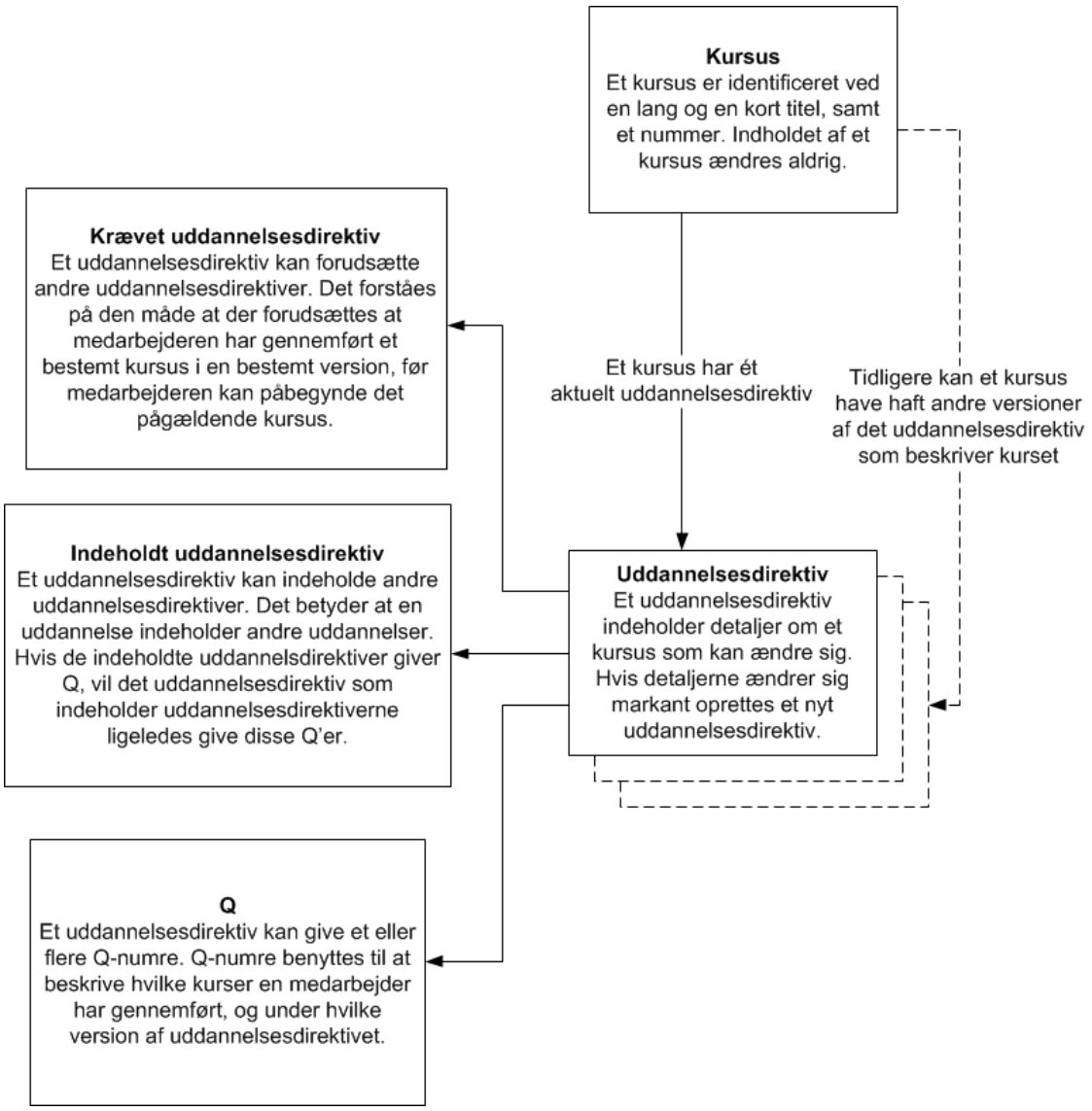
Hvis et kursus indeholder et andet kursus, vil på på konceptuelt plan sige at uddannelsesdirektivet indeholder uddannelsesdirektiverne for de kurser som er indeholdt.

Et eksempel på en udddannelse som indeholder andre uddannelser er "Grunduddannelse komplet". Denne uddannelse indeholder 11 mindre uddannelser, som også kan tages separat. Se nedenstående diagram:



FIGUR 4 - GRUNDUDDANNELSE KOMPLET INDEHOLDER ANDRE KURSER

I afsnittet "Fagplaner og dokumenter" vil vi beskrive i detaljer hvordan indhold som er delt mellem kurser i form af fagplaner og dokumenter, håndteres.



FIGUR 5 - RELATION MELLEM KURSER OG UDDANNELSESDIREKTIVER

Vi har altså at et kursus, som det opfattes af en medarbejder, i datamodellen består at to entiteter, nemlig *kursus* og *uddannelsesdirektiv*. Kursus indeholder basale informationer som ikke vil ændre sig, så som navnet på kurset, mens uddannelsesdirektiv indeholder alle detaljer om kurset, så som kursets indhold og forudsætninger. Et uddannelsesdirektiv kan ændres, men hvis det i markant grad ændrer indhold, vil det ofte være sådan, at der i stedet oprettes et nyt, med udgangspunkt i det gamle, men som giver et andet Q-nummer. Idet Q-nummeret benyttes til at identificere hvilke kurser en medarbejder har gennemført, og i hvilke versioner, har hjemmeværnet på den måde mulighed for at sikre at en potentiel deltager har gennemført et bestemt forudsat kursus, i en aktuel version.

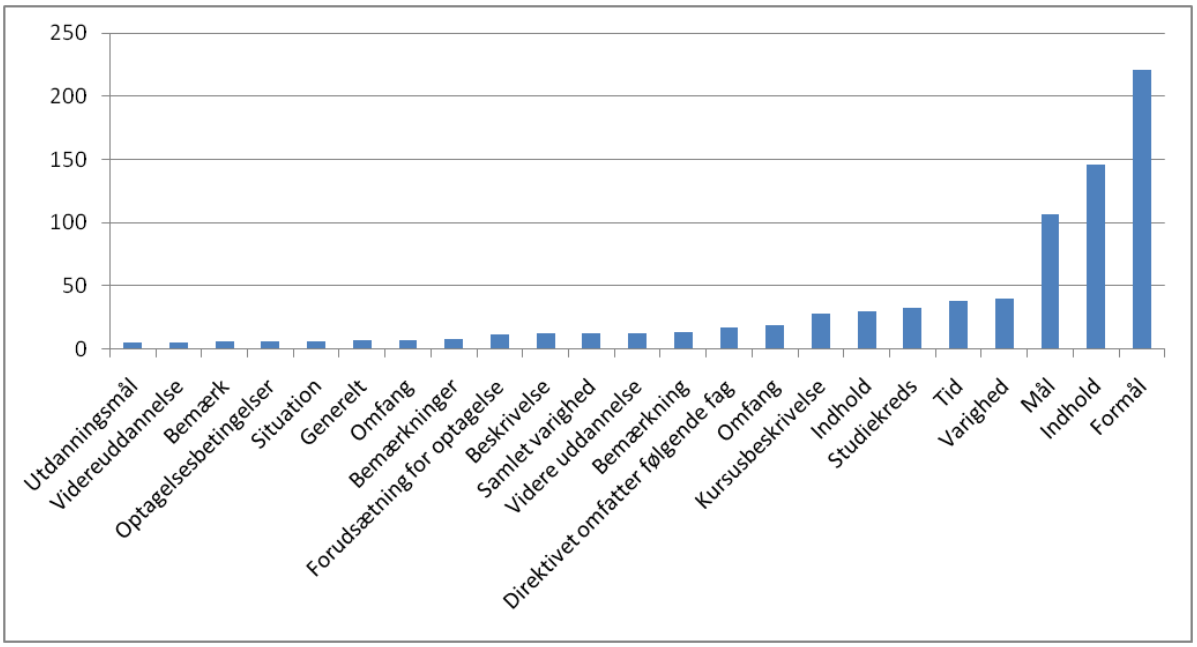
UDDANNELSESDIREKTIVER

Et uddannelsesdirektiv indeholder detaljeret information omkring en bestemt uddannelse, så som:

- Formål
- Mål
- Indhold
- Tid
- Fagplaner
- Dokumenter

Hjemmeværnet udbyder pt. omkring 400 uddannelser som har et uddannelsesdirektiv. Indhold og struktur af uddannelsesdirektiver varierer markant. Da Hjemmeværnet ønsker at være i stand til at repræsentere uddannelsesdirektiver på samme måde som de gør det i øjeblikket, har vi foretaget en analyse af de eksisterende uddannelsesdirektiver, med henblik på at afgøre hvad vores datastruktur bør kunne indeholde.

Detaljerede resultater og fremgangsmåde for analysen findes i appendix under "Punkter i uddannelsesdirektiver", men vores konklusion er følgende: Der findes over 50 forskellige punkter i uddannelsesdirektiverne, og de bærer præg af at de ikke har været indtastet i en formular med faste felter. Nedenfor ses et diagram over de mest benyttede punkter i uddannelsesdiagrammerne:



FIGUR 6 - HYPPIGHEDEN AF DIVERSE PUNKTER I UDDANNELSESDIREKTIVER

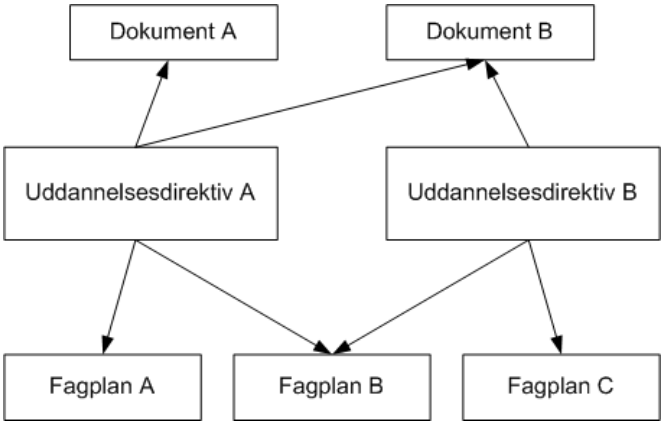
Som det ses af ovenstående diagram er det felterne "mål", "formål", "indhold" og "tid"/"varighed" der benyttes mest - De fremgår i mindst 20% og op til over 50% af uddannelsesdirektiverne. Bemærk at punkterne "tid" og "varighed" benyttes til samme type information.

Derfor har vi valgt at vores datastruktur skal indeholde netop disse fire faste punkter, hvorimod de resterende punkter skal gemmes i et felt ved navn "andet" i vores datastruktur. Feltet "andet" vil tillade HTML, derfor vil man kunne opnå samme formattering som der haves nu.

Fordelen ved at begrænse antallet af faste felter, er at hvis man eksempelvis har 50 forskellige felter vil det være besværligt for brugeren at finde netop det felt han ønsker at taste i, og desuden er der stor sandsynlighed for at det jævnligt vil vise sig at der ønskes et bestemt felt som datastrukturen ikke vil understøtte, fordi det ikke er blevet brugt tidligere. Ulempen ved et lavt antal felter, hvor man indsætter alle de resterende punkter i en "andet"-kategori er at dataen derved bliver mindre struktureret, hvilket bl.a. gør det mere vanskeligt at foretage specifikke søgninger i uddannelsesdirektiver.

FAGPLANER OG DOKUMENTER

Visse uddannelsesdirektiver har udover deres umiddelbare indhold også tilknyttet fagplaner og andre dokumenter. Uddannelsesdirektiver kan dele fagplaner og dokumenter, se eksempelvis nedenstående illustration:



FIGUR 7 - RELATIONER MELLEM DOKUMENTER, FAGPLANER OG UDDANNELSESDIREKTIVER

Fagplaner er at opfatte som en slags "mini-uddannelsesdirektiver" for dele af en uddannelse, hvis uddannelsen har et omfattende indhold.

En fagplan har delvist struktureret indhold, og indeholder på tilsvarende vis som for uddannelsesdirektiver punkter som "mål", "formål", "indhold" og "tid". Derudover findes der en række mindre benyttede punkter der ligesom for uddannelsesdirektiver, placeres som formatteret tekst under "andet" i datastrukturen. Desuden har en fagplan et nummer og en dato som indikerer hvornår den er oprettet.

Et eksempel på dette er kurset "Politifunktionskursus Almindelig Hjælp", som indeholder en fagplan for "Tjenestekendskab". "Tjenestekendskab" er ikke i sig selv en uddannelse, men er en beskrivelse af en del af pensum som er fælles for flere uddannelser, men dette pensum kan ikke tages for sig selv, og giver ikke noget Q. "Gruppeførerkursus, Politi" indeholder ligeledes fagplanen "Tjenestekendskab".

Dokumenter er ustruktureret information om et kursus. Det kan bl.a. være beskrivelser af øvelser som hører til en uddannelse. Disse dokumenter kan deles af flere uddannelser. Et eksempel på dokumenter er følgende 3 dokumenter: "Samarbejdsøvelse 1", "Samarbejdsøvelse 2" og "Samarbejdsøvelse 3". Disse dokumenter henvises der blandt andet til, fra kurset "RLU, Småstyrkes Kamp 1".

TILDELING AF Q

De fleste uddannelser giver et eller flere Q-numre. Q-nummeret benyttes af kursusdeltagere til at identificere uddannelser der er taget. Uddannelser benytter også Q-numre til at vise hvilke uddannelser der er forudsat af uddannelsen. I praksis omtales det, at man har gennemført en bestemt uddannelse, internt i hjemmeværnet som at "man har opnået Q". Modsat hvad man måske kunne tro, har Q ingen skjult betydning som man kan "dekode" for at finde information om kursus eller versionsnummer, men er ganske enkelt et fortløbende tal fra en sekvens som der trækkes fra, når et uddannelsesdirektiv opdateres.

Grundlæggende er der tre muligheder for hvordan en bestemt uddannelse kan forholde sig til Q:

1. Uddannelsen giver intet Q.
2. Uddannelsen giver ét Q-nummer.
3. Uddannelsen giver mere end et Q-nummer.

UDDANNELSER SOM IKKE GIVER Q

Dette er generelt specielle kurser som ikke rigtig passer ind i resten af uddannelsessystemet, og som ikke direkte relaterer sig til øvrige kurser. Eksempler på denne type kurser er "Landsrådsmøde", "Studie- og udviklingsseminar" og "Introduktionskursus for militært ansat personale".

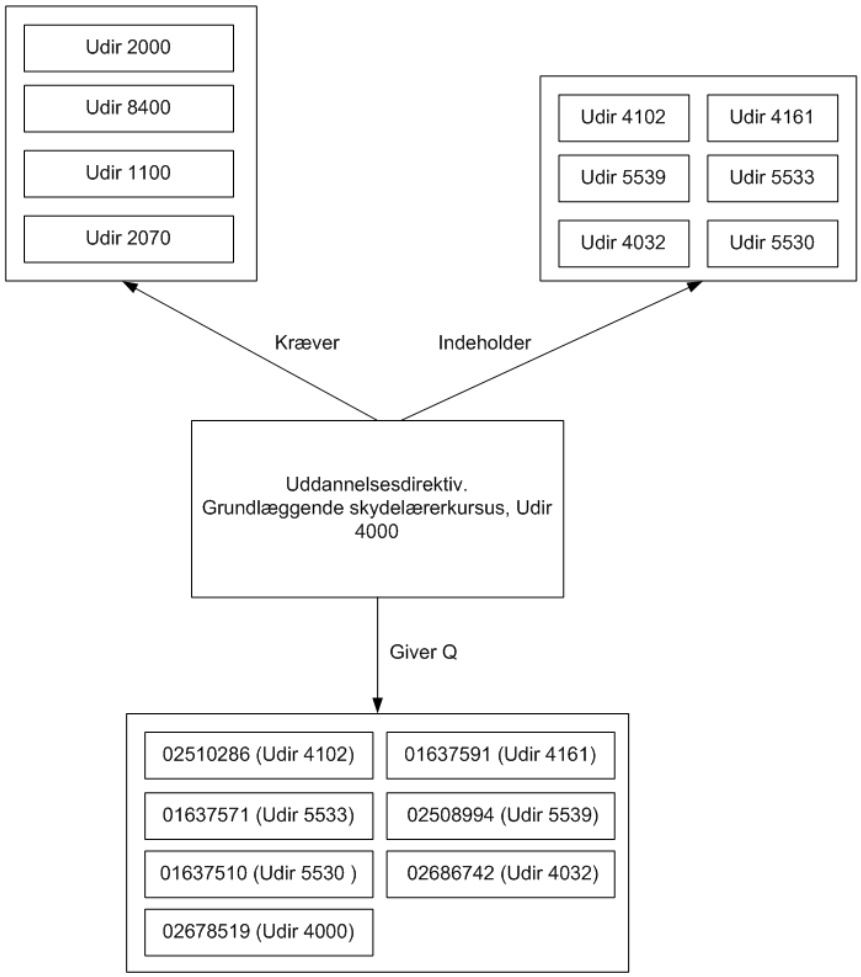
UDDANNELSER SOM GIVER ÉT Q-NUMMER

Dette er klart det mest almindelige for Hjemmeværnets kurser. Eksempler på uddannelser som kun giver et Q er: "Hundeførerfunktionskursus", "Signalofficerskursus" og "Sprængningshjælperkursus".

UDDANNELSER SOM GIVER MERE END ET Q-NUMMER

Der er en rækker eksempler på kurser som giver mere end et Q-nummer. Der kan være tre mulige årsager til dette:

1. Uddannelsen har indhold som overlapper med indholdet på et andet kursus, i en sådan grad at kurset skal give det Q som man ellers ville få for at tage den anden uddannelse. Dette skal ikke forveksles med uddannelser som rent faktisk *indeholder* andre uddannelser.
2. Uddannelsen indeholder andre uddannelser. Et udmærket eksempel på dette er "Grundlæggende skydelærerkursus". Nedenfor findes en illustration af hvor det fremgår hvorfor netop denne uddannelse giver flere Q-numre.
3. Uddannelsen har ikke indhold som overlapper andre eksisterende uddannelser, og indeholder heller ikke andre separate udddanelser. Der kan være forskellige årsager til at uddannelser er kommet i denne tilstand, oftest vil det være fordi uddannelsen har haft overlappende indhold med en uddannelse som nu ikke længere findes, eller som nu giver et andet Q-nummer. Denne situation er derfor opstået af historiske årsager, og det bør altid kunne ændres således at uddannelsen falder i en af de to foregående kategorier, eller således at uddannelsen kun giver et enkelt Q. Det er dog nødvendigt at vores system understøtter dette særtilfælde.



FIGUR 8 - GRUNDLÆGGENDE SKYDELÆRERKURSUS MED UDIR 4000 GIVER 7 FORSKELLIGE Q'ER, HVOR DE 6 AF DEM KOMMER FRA ANDRE RELATEREDE UDDANNELSER SOM ER INDHOLDT I UDDANNELSEN, MENS DET SYVENDE KOMMER FRA UDDANNELSEN SELV

OPDATERING AF KURSER

Det er et centralt krav fra Hjemmeværnets uddannelsessektion at de får mulighed for at opdatere indholdet af eksisterende kurser. Alle kurser af betydning giver et eller flere Q-numre, der benyttes som en identifikation af hvilke kurser en bestemt medarbejder har gennemført. Fundamentale ændringer i et kursus kan gøre at kurset giver andre Q-numre end tidligere. Dette kan opfattes som at kurset nu findes i en ny version. Ændringer til kurser kan inddeles i følgende kategorier:

1. Ændringer af titlen for et bestemt kursus.
2. Mindre ændringer i indholdet for et kursus, for eksempel ændringer af kursets tilrettelæggelse, hvor ændringen ikke skal resultere i et nyt Q-nummer.
3. Fundamentale ændringer i kurset som gør at kurset skal have et nyt Q-nummer

Det viser sig at man i praksis aldrig ændrer i titlen for et kursus. Det skyldes at hvis man mener at kurset har ændret sig i en så markant grad at det kræver en ændring af titlen så skal der i stedet oprettes et helt nyt kursus.

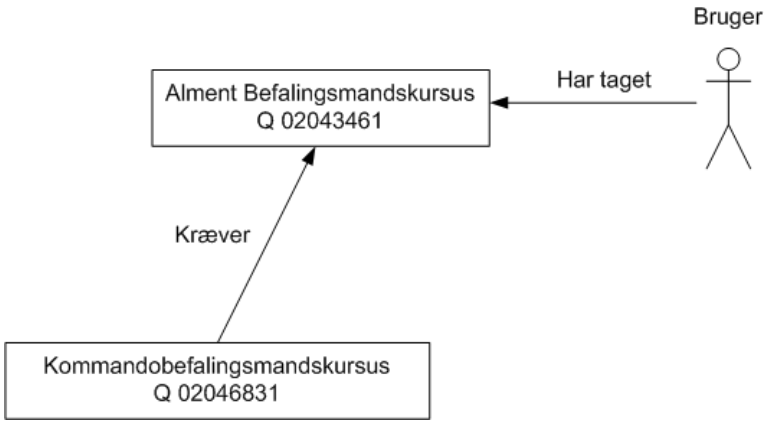
Vores model for kurser inddeler konceptet "Et kursus" i to separate entiteter:

- kursus
- udir (Uddannelsesdirektiv)

For ét bestemt kursus kan der findes flere uddannelsdirektiver. Der kan dog kun findes ét som er det aktuelle. I situationen hvor der skal foretages mindre ændringer på kurset foretages disse ændringer på den aktuelle udir-entitet, uden at det har yderligere konsekvenser for databasen. Foretages der derimod fundamentale ændringer i kursets aktuelle uddannelsesdirektiv kan det kræves at kurset får et helt nyt uddannelsesdirektiv. I den situation vil man oftest også ønske at skifte hvilke Q-numre der efterfølgende gives når kurset gennemføres. Når dette sker har det konsekvenser for andre dele af databasen:

1. Brugere som har taget det kursus (Q-nummer), som nu erstattes af et nyt uddannelsesdirektiv.
2. Kurser som forudsætter det kursus (uddannelsesdirektiv), som nu erstattes af et nyt uddannelsesdirektiv

Følgende situation diagram illustrerer situationen:



FIGUR 9 - KOMMANDOBEFALINGSMANDSKURSUS FORUDSÆTTER ALMENT BEFALINGSMANDSKURSUS,

SOM BRUGEREN HAR GENNEMFØRT

Man forestiller sig nu at uddannelsesdirektivet for "Alment Befalingsmandskursus" nu er blevet væsentligt ændret og har fået nyt aktuelt uddannelsesdirektiv. Dette uddannelsdirektiv giver Q 02043462.

KONSEKVENS FOR BRUGERE

Kommandobefalingsmandskursus kræver nu Q 02043462, hvilket vil sige at brugere som har taget Q 02043461 nu skal tage kurset Q 02043462 før de kan få adgang til Q 02046831.

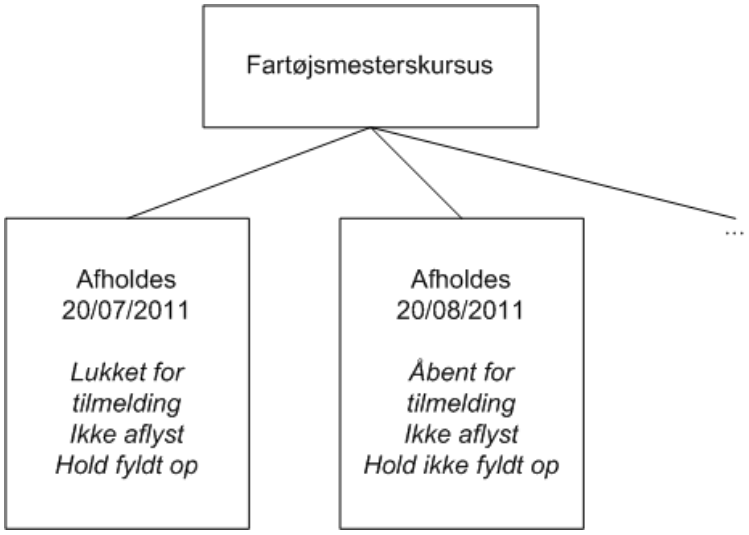
I situationen hvor der kræves kursus Q 02043462 vil der formelt set kræves at potentielle deltagere på kurset tager hele dette nye kursus, selvom de eventuelt allerede har gennemført den gamle version af kurset Q 02043461. Her vil det bero på en individuel vurdering fra den uddannelsesansvarlige at give merit for de dele af kurset som eventuelt er fælles med den gamle uddannelse til den potentielle kursist. Dette vil foregå udenom uddannelsesstyringssystemet.

KONSEKVENS FOR ANDRE UDDANNELSER

Systemet vil automatisk gå alle uddannelser igennem som forudsætter det gamle uddannelsesdirektiv og opdatere disse uddannelser så de i stedet forudsætter det nye uddannelsesdirektiv.

AFHOLDELSE AF ET KURSUS

Der er behov for at kunne repræsentere en bestemt instans af et kursus, det vil sige en beskrivelse af at et bestemt kursus afholdes på en bestemt dato. Desuden har man brug for at kunne repræsentere detaljer om en bestemt instans af et kursus, så som om det er fyldt eller aflyst, eller om det er lukket for tilmelding. En illustration af dette ses nedenfor:



FIGUR 10 - KURSUSINSTANSER AF FARTØJSMESTERSKURSUS

Når en bruger bliver tilmeldt et kursus foregår det ved at der skabes en relation mellem en bestemt bruger, og en bestemt kursusinstans. Denne relation har desuden en boolsk værdi som indikerer om kurset er bestået. Denne vil blive sat til sand når kurset er bestået.

Der er behov for at registrere følgende information om en kursusinstans:

- Sted
- Kursusstatus
- Startdato
- Slutdato
- Skole

Hver gang et kursus afholdes, angives et sted og en skole. Dette kunne eksempelvis være Skive kaserne, UDC MJY. Der angives både en startdato og en slutdato for en bestemt kursusafholdelse - For kurser som kun varer én dag, angives samme dato i start og slut. Applikationen kan så vælge ikke at vise slutdatoen.

Et kursus kan være i et endeligt antal tilstande, som beskrives af kursusstatus. Disse tilstande er følgende:

- Åbent for tilmelding
- Kursus afholdt
- Kursus lukket for tilmelding
- Kursus fyldt op
- Kursus afholdes ikke
- Kursus aflyst
- Kursus aflyst på grund af manglende tilmeldinger

BRUGERE I SYSTEMET

Vi ønsker at repræsentere alle de informationer om en bruger som Hjemmeværnet almindeligvis registrerer om sine medarbejdere. Dette er dels almindelige informationer så som telefonnummer, navn og adresse, men er også information om de køretøjer som medarbejderen ejer. Informationer som hvilke køretøjer en bruger har rådighed over, har ikke umiddelbart relevans for systemet, hvis det udelukkende anses for at være et kursusstyringssystem, men da vi ønsker at lade systemet på komplet vis repræsentere de to overordnede entiter i systemet, nemlig medarbejdere og kurser, er det nødvendigt også at have disse informationer.

Der registreres følgende personlige informationer om en medarbejder:

- MA-nummer (Medarbejdernummer)
- CPR-nummer
- Fornavn
- Efternavn
- Adresse
- Postnummer
- Telefonnummer
- Mobilnummer
- Email
- Køretøj

Vi registrerer information om køretøjet for sig selv, da en medarbejder kan have flere forskellige køretøjer, mens de resterende informationer registreres på selve bruger-entiteten.

BRUGERE OG ORGANISATIONEN

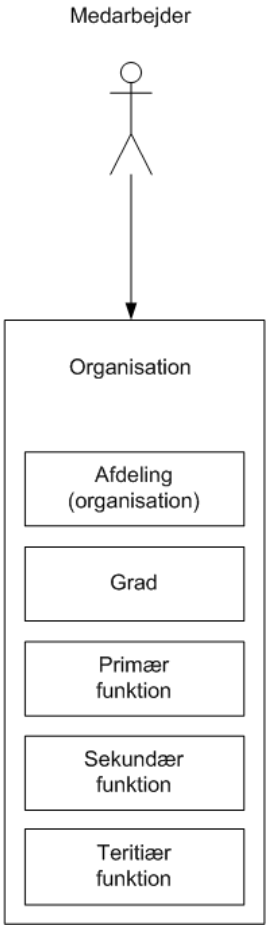
Vi har desuden behov for at relatere medarbejderen til organisationen, for at kunne vise eksempelvis hans grad og funktion. I forhold til organsationen registrerer vi følgende informationer om brugeren:

- Medarbejdergruppe (Om brugeren er fx frivillig eller fuldtidsansat)
- Grad
- Organisation
- Primær funktion
- Sekundær funktion
- Tertiær funktion

En medarbejder vil som illustrere, være tilknyttet en organisation. Dette kunne fx være "Hjemmeværnskommandoen, Hærhjemmeværnet, Totalforsvarsregion Sjælland".

Indenfor denne organisation kan man have op til tre funktioner indenfor Hjemmeværnet. Disse funktioner forudsætter almindeligvis at en bestemt uddannelse er gennemført. I vores datamodel vil vi understøtte et vilkårligt antal funktioner for hver bruger. Til hver funktion vil vi tilknytte en prioritet. Dermed lader vi det være op til applikationen at sikre at der maksimalt registreres tre funktioner for en bruger - Dermed skal datamodellen ikke ændres hvis man ønsker at understøtte et andet antal funktioner per bruger, hvilket gør den mere fleksibel.

Desuden kan en medarbejder også relateres til en afdeling i organisationen hvis medarbejderen er leder for afdelingen.



FIGUR 11 - MEDARBEJDER OG ORGANISATION

BRUGERE OG UDDANNELSER

I forhold til kurser registrerer vi op til tre informationer om brugeren:

1. Kurser som brugeren har deltaget på
2. Kurser som brugeren er kursusansvarlig for (Kun for brugere som er kursusansvarlig)
3. Kususinstanser som brugeren står for at afholde (Kun for brugere som er kursusansvarlige og afholder et bestemt kursus)

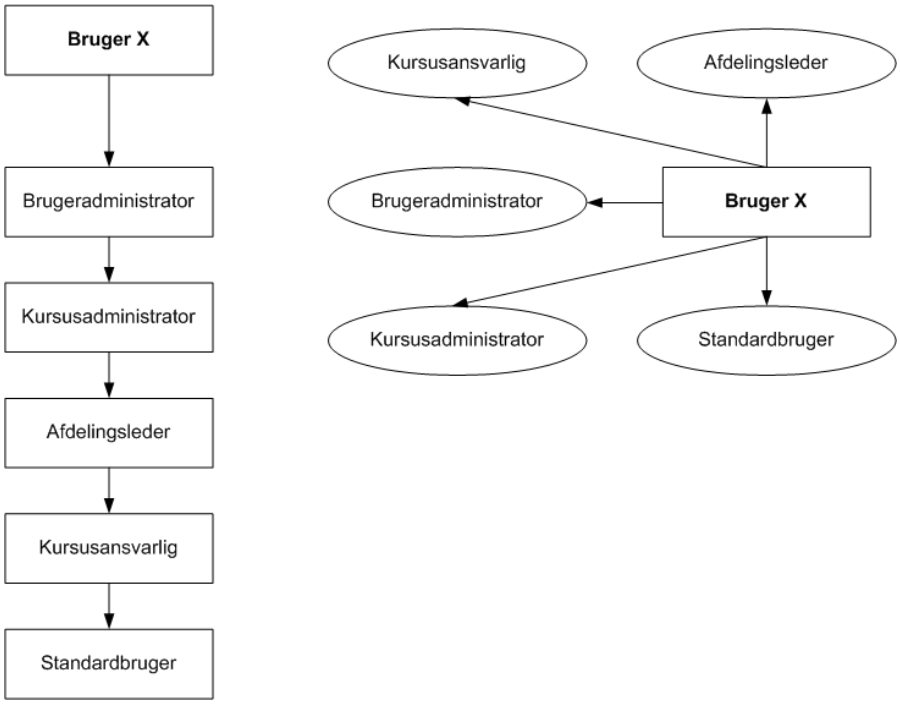
Gennem de kurser som brugeren har deltaget på, kan man afgøre hvilke Q'er som brugeren har taget, samt alle øvrige detaljer om det kursus som brugeren deltog på.

BRUGERE OG RETTIGHEDER

Under udarbejdelse af kravspecifikationen er vi nået frem til at der indenfor systemet findes følgende roller som en bruger kan have:

- Menige ansatte og frivillige i Hjemmeværnet (Brugere).
- Afdelingsledere
- Kursusansvarlige
- Kursusadministratorer
- Brugeradministratorer

Vi har besluttet at basere systemets rettighedsstyring på roller frem for et bruger-hierarki. I forhold til de bruger-typer som der er behov for, vil en brugerhierarki-løsning muligvis være acceptabel, men det bliver en del mere fleksibelt hvis vi baserer det på roller. Nedenfor findes en illustration af hvordan et brugerhierarki-baseret rettighedssystem vil se ud for en bestemt bruger, set i forhold til et rolle-baseret hierarki:



FIGUR 12 - FORSKELLEN PÅ BRUGERHIERARIKI (T.V) OG ROLLEBASERET RETTIGHEDSSYSTEM (T.H)

I et brugerhierarki vil en bruger altså automatisk få alle rettigheder som brugere lavere i hierarkiet har, hvorimod at man i et rolle-baseret rettighedssystem vil give én rolle af gangen. Hvad der er optimalt vil afhænge af om rettighederne kan ordnes i en lige linje, hvor der aldrig vil være behov for at give en bruger en rettighed, uden at vedkommende også bør have eventuelle underordnede rettigheder, eller om rettighederne er mere uafhængige.

Illustrationen indeholder de rettigheder som der er behov for i systemet. Det er ikke givet at en kursusansvarlig skal være afdelingsleder eller omvendt, derfor benyttes et rolle-baseret rettighedssystem.

ROLLER

Når det er muligt vil vi benytte en bestemt datastruktur til roller, som vi vil relatere brugere til, hvis de har hver enkelt rolle. Der er dog to roller hvor dette ikke vil være tilstrækkeligt, nemlig afdelingsleder og kursusansvarlig.

For en afdelingsleder er det ikke tilstrækkeligt at registrere om medarbejderen er afdelingsleder, men også hvor medarbejderen er leder. Da en medarbejder allerede tilhører en eller flere afdelinger kunne det umiddelbart virke oplagt at registrere lederrollen direke på denne relatation, men da en bruger godt kan være leder på et højere organisationsniveau end det som brugeren selv er tilknyttet, vil denne løsning ikke fungere. Derfor benyttes en separat relation til at indikere hvad en bestemt bruger er leder for, og det er tilstedeværelsen af denne relation som er afgørende for rollen “Afdelingsleder”.

For en kursusansvarlig skal rollen på tilsvarende vis afgøres ved om brugeren er relateret til et bestemt kursus, ligesom det skal være muligt for brugeren at være ansvarlig for flere kurser. Derudover bliver en kursusansvarlig automatisk ansvarlig for en bestemt kursusinstans, når han opretter kursusinstansen. Det giver vedkommende retten til at ændre kursusstatus for kurset, samt for at bestå kursisterne efter at kurset er gennemført.

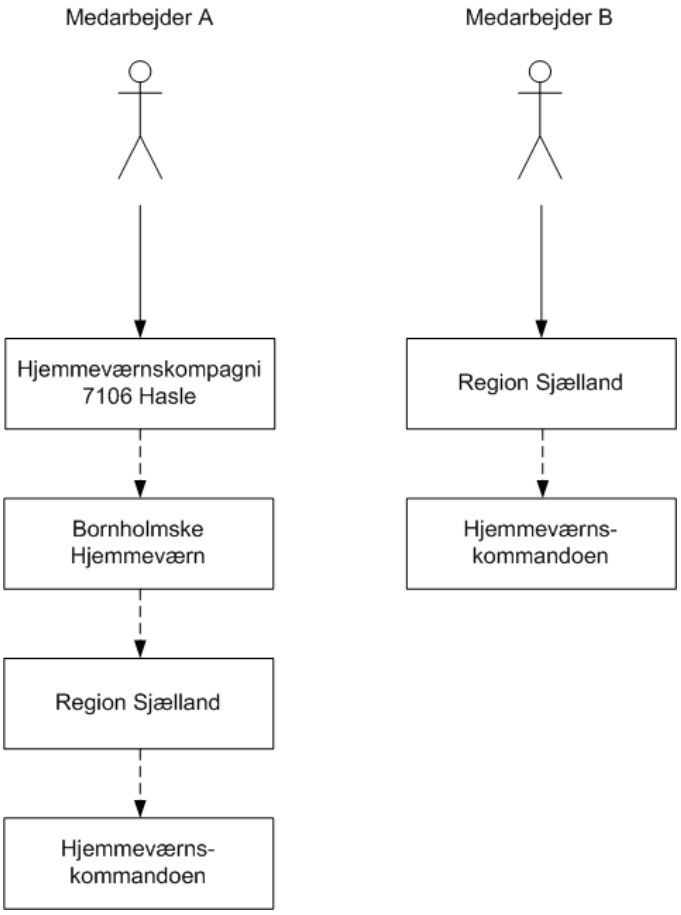
ORGANISATION

Hjemmeværnets organisation er opdelt i fire trin, i det følgende vil vi med en samlende betegnelse, omtale trinene som afdelinger. De fire afdelinger er hierarkisk opbygget efter følgende model:

1. Hjemmeværnskommandoen
2. Regioner
3. Distrikter
4. Underafdelinger

En bestemt medarbejder kan fx være tilknyttet Hjemmeværnskompagni 7106 Hasle under Det Bornholmske Hjemmeværn, Region Sjælland, Hjemmeværnskommandoen. Det vil altid være tilstrækkeligt at nævne den afdeling som medarbejderen er tilknyttet, der ligger lavest i hierarkiet. Med den laveste afdeling vil man entydigt kunne afgøre hvilke afdelinger man tilhører højere oppe i hierarkiet, da hver afdeling er underafdeling af netop én anden afdeling, med undtagelse af Hjemmeværnskommandoen, som er øverst i hierarkiet.

Vi har dog også at en medarbejder kan være knyttet til en afdeling i organisationen som er højere oppe i hierarkiet, eksempelvis Region Sjælland. Et eksempel på dette ses nedenfor:



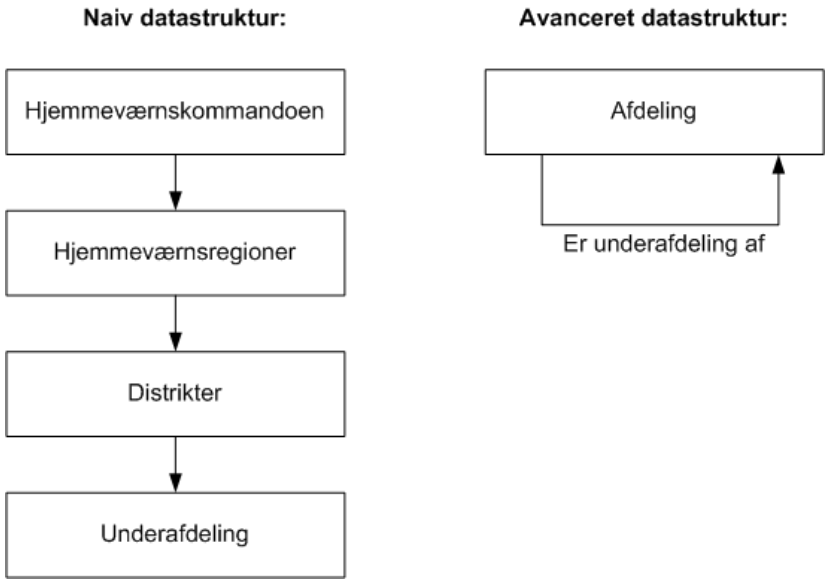
FIGUR 13 - MEDARBEJDER A ER KNYTTET TIL UNDERAFDELINGEN 'HJEMMEVÆRNSKOMPAGNI 7106 HASLE',

MENS MEDARBEJDER B ER KNYTTET TIL REGION SJÆLLAND

Dette kan give nogle vanskeligheder i datamodellen når informationen skal struktureres, fordi man ønsker at relatere brugeren til den afdeling vedkommende tilhører, men for forskellige medarbejdere kan det være forskellige trin i hierarkiet, der skal relateres til.

Forestillede man sig at man fra medarbejderens side henviste til den afdeling som var nederst i hierarkiet, ville det være nødvendigt at man i datastrukturen havde mulighed henvise til en afdeling på et vilkårligt trin i hierarkiet. Det ville som udgangspunkt kræve én potentiel relation til hver af de fire afdelinger, hvoraf kun den ene havde mening, nemlig den som var til det laveste trin som kunne påføres den pågældende medarbejder. En sådan løsning ville dog give en uheldig datastruktur, fordi man ville have en gruppe på fire felter, hvor det altid kun var den ene som var i brug, hvilket ville give spildplads i databasen på grund af NULL-rækker.

For at imødekomme denne uhensigtsmæssighed har vi valgt at håndtere alle afdelinger i den samme datastruktur, som der så henvises til fra brugeren, for at indikere hvilken afdeling medarbejderen er tilknyttet. Dette giver umiddelbart en mere kompliceret datastruktur, end hvis man bare havde fire felter, fordi det nu er datastrukturens indhold som afgør hierarkiet, frem for datastrukturen i sig selv. Nedenfor findes en illustration som viser den naive struktur man kunne have valgt, men som ville give problemer i forhold til relationerne, samt den avancerede struktur som vi har valgt.



FIGUR 14 - DE TO LØSNINGSFORSLAG TIL AFDELINGSRELATIONER

Udover at udgå NULL-rækker, giver den avancerede datastruktur også en mere fleksibel måde at repræsentere hierarkiet og dermed organisationens struktur, da man ikke er begrænset til præcis fire afdelinger - Man kan på et senere tidspunkt indskyde eller fjerne en anden afdelingstype.

ORGANISATIONENS ROLLE I DATAMODELLEN

Det er nødvendigt at organisationen repræsenteres på en forholdsvis indgående måde i datamodellen, selvom det er brugere og uddannelse uddannelsessystemet har fokus på. Dette skyldes til dels at vi ønsker at beskrive

medarbejdernes egenskaber i forhold til organisationen, herunder informationer som medarbejderens funktion, medarbejdergruppe og grad.

Derudover skaber organisationen en binding mellem medarbejdere og uddannelser i forhold til medarbejderens funktion.

MEDARBEJDERENS FUNKTION

De fleste funktioner man kan bestride i Hjemmeværnet, kræver at man har bestået én eller flere uddannelser. Eksempelvis kan nævnes den funktion som ved sit korte navn betegnes KDOBM, kommandobefalingsmand. For at kunne bestride denne funktion er det krævet at medarbejderen tager Q 02046831 fra kurset Kommandobefalingsmandskursus.

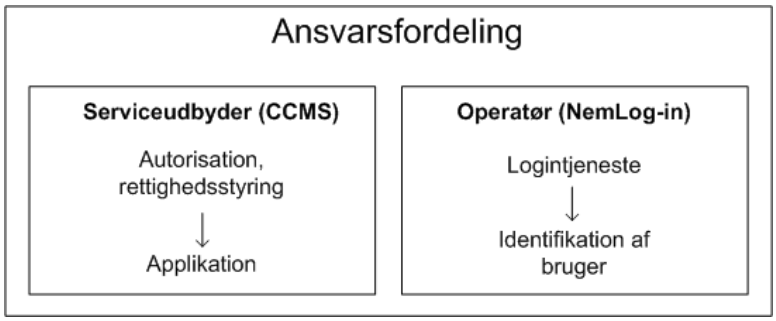
Det kan i den sammenhæng også have relevans for visse brugere af systemet at kunne slå op hvilke kurser en bestemt medarbejder skal gennemføre, for at kvalificere sig til en bestemt funktion.

AUTENTIFIKATION

Det er et krav at bruger autentificeres gennem NemID. For Hjemmeværnet vil det betyde at der findes to mulige løsninger:

1. Tilslutning via NemLog-in eller Virk.dk
2. Tilslutning direkte i forhold til DanID

Løsning 1 er mulig fordi Hjemmeværnet er en offentlig institution[1]. Denne løsning er at foretrække fordi, fordi den giver adgang til den fællesoffentlige single sign-on løsning.



FIGUR 15 - ANSVARSFORDELING MELLEM CCMS OG NEMLOG-IN

Ovenfor findes en illustration som beskriver hvilken rolle NemLog-in ville spille i forhold til CCMS.

Det er vigtigt at bemærke at dette afsnit om autentifikation gennem NemLog-in skal opfattes som en foreløbig analyse og beskrivelse, da dette ikke er hvor hovedvægten af projektet findes. En komplet analyse af detaljerne

[1] http://danid.dk/download/tu9/Introduktion%20til%20NemID%20og%20Tjenesteudbyderpakken.pdf

omkring integration af en offentlig webapplikation mod NemLog-in, med risikoanalyse, beredskabsplaner samt implementationsspecifikke overvejelser, udgør i sig selv et større projekt.

Formålet med denne indledende analyse har primært været at sikre at systemet er designet på en måde, så det umiddelbart understøtter NemLog-in via NemID.

Den indledende analyse viser at det vil være nødvendigt at have grundlag for at kunne oversætte en persons identitet fra personens ID indenfor SAML-systemet til brugerens ID indenfor CCMS.

Indenfor NemLog-in benyttes OCES-attributterne CPR, PID og RID til at identificere en person[2]. Ved hjælp af direkte account mapping kan vi oversætte en person som identificeret gennem NemLog-in, til en bruger i vores system, da Hjemmeværnet ligeledes gemmer CPR-nummeret for deres medarbejdere. Derfor vurderer vi umiddelbart at de informationer som der er behov for, for at kunne benytte NemLog-in, er til stede.

ENTITETER

GENNEMGANG AF DIAGRAM

Databasediagrammet er inddelt i 3 overordnede områder:

- Personel
- Uddannelse
- Organisation

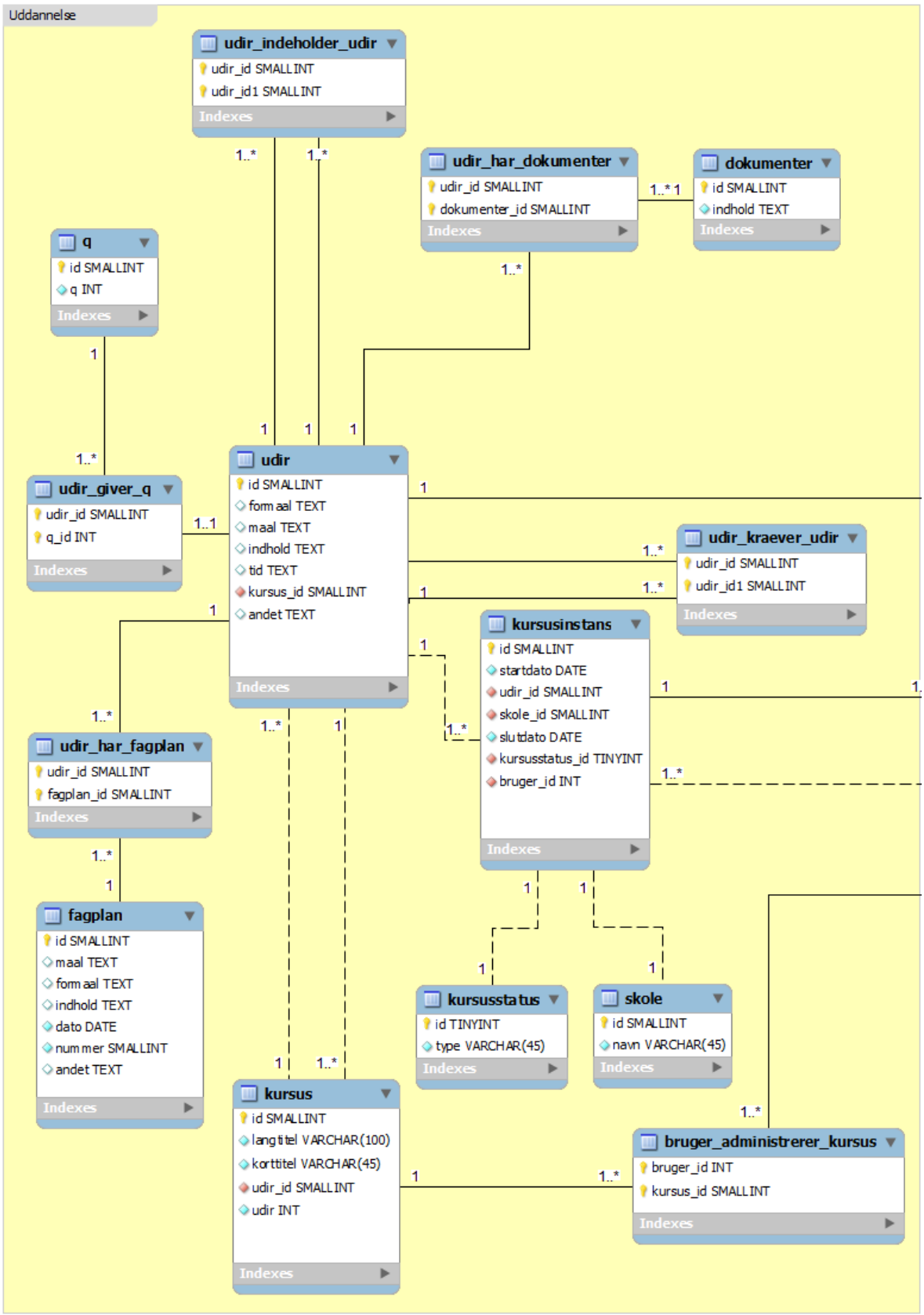
Inddelingen er med til at simplificere diagrammet ved at skabe en logisk gruppering af entiteterne. I appendix findes det samlede diagram. I de følgende afsnit vil vi vise relevante udsnit af diagrammet.

UDDANNELSE

Følgende entiteter udgør uddannelses-sektionen i databasediagrammet:

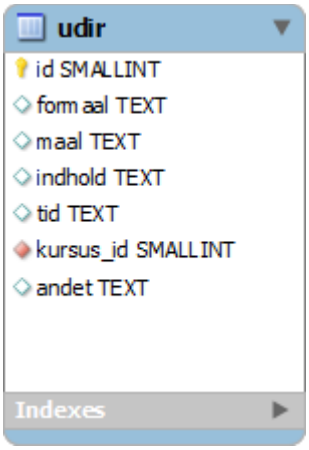
- udir
- udir_indeholder_udir
- udir_giver_q
- q
- udir_har_dokumenter
- dokumenter
- udir_kraever_udir
- udir_har_fagplan
- fagplan
- kursus
- bruger_administrerer_kursus
- kursusinstans
- kursusstatus
- skole

[2]Guide til Føderaltilslutning V 1.1 - http://www.skat.dk/getFile.aspx?Id=58681



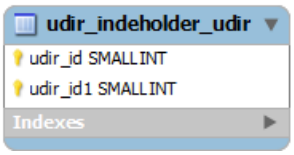
FIGUR 16 - DATABASEDIAGRAM OVER UDDANNELSE

UDIR



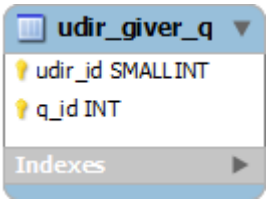
Primærnøgle(r):	id *(SMALLINT)* Autonummereret identifikationsnummer.
Fremmednøgle(r):	kursus_id *(SMALLINT)* Fremmednøgle til kursus
Attributter:	formaal *(TEXT)* Kursets formål maal *(TEXT)* Kursets mål indhold *(TEXT)* Kursets indhold tid *(TEXT)* Kursets tidsmæssige omfang andet *(TEXT)* Feltet *andet* indeholder de punkter fra uddannelsesdirektivet som ikke passer i de øvrige kategorier. Feltet indeholder formatteret tekst.

UDIR_INDEHOLDER_UDIR



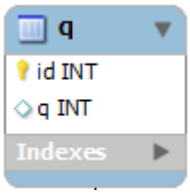
Primærnøgle(r):	udir_id *(SMALLINT)* Id for uddannelsesdirektivet som denne relation tager udgangspunkt i udir_id1 *(SMALLINT)* Uddannelsesdirektivet med udir_id indeholder dette uddannelsesdirektiv, udir_id1

UDIR_GIVER_Q



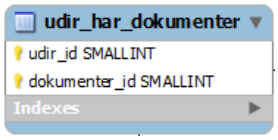
Primærnøgle(r):	udir_id *(SMALLINT)* Uddannelsesdirektivets id q_id *(INT)* Q som uddannelsesdirektivet giver ved fuldførelse

Q



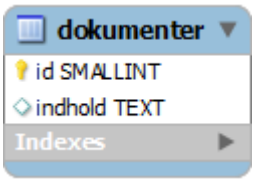
Primærnøgle(r):	id *(INT)* Autonummereret identifikationsnummer
Attributter:	q *(unik, ikke null) - (INT)* Det pågældende q-nummer

UDIR_HAR_DOKUMENTER



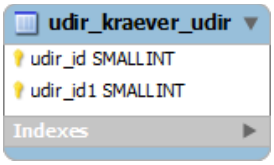
Primærnøgle(r):	udir_id *(SMALLINT)* Uddannelsesdirektivets id dokumenter_id *(SMALLINT)* Id for det dokument som er tilknyttet det pågældende uddannelsesdirektiv.

DOKUMENTER



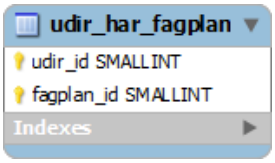
Primærnøgle(r):	id *(SMALLINT)* Autonummereret identifikationsnummer
Attributter:	indhold *(ikke null) - (TEXT)* Dokumenter indeholder ustruktureret data, derfor findes hele indholdet i en enkelt attribut. Denne attribut kan indeholde formateret tekst.

UDIR_KRAEVER_UDIR



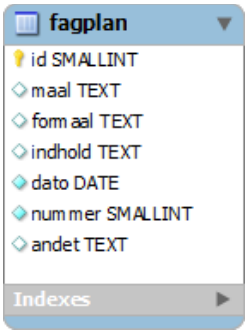
Primærnøgle(r):	udir_id *(SMALLINT)* Id for uddannelsesdirektivet som denne relation tager udgangspunkt i udir_id1 *(SMALLINT)* Uddannelsesdirektivet med udir_id kræver dette uddannelsesdirektiv, udir_id1

UDIR_HAR_FAGPLAN



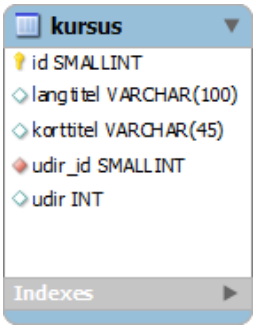
Primærnøgle(r):	udir_id *(SMALLINT)* Uddannelsesdirektivets id fagplan_id *(SMALLINT)* Id for den fagplan som er tilknyttet det pågældende uddannelsesdirektiv.

FAGPLAN



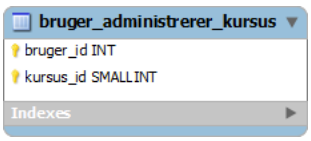
Primærnøgle(r):	id *(SMALLINT)* Autonummereret identifikationsnummer.
Attributter:	maal *(TEXT)* Fagplanens mål formaal *(TEXT)* Fagplanens formål indhold *(TEXT)* Fagplanens indhold dato *(ikke null) - (DATE)* Dato for fagplanens oprettelse nummer *(unik, ikke null) - (SMALLINT)* Fagplaner er nummererede dokumenter. andet *(TEXT)* Feltet *andet* indeholder de punkter fra fagplanen som ikke passer i de øvrige kattegorier. Feltet indeholder formatteret tekst.

KURSUS



Primærnøgle(r):	id *(SMALLINT)* Autonummereret identifikationsnummer
Fremmednøgle(r):	udir_id *(SMALLINT)* Fremmednøgle til uddannelsesdirektivet
Attributter:	langtitel *(unik, ikke null) - (VARCHAR(100))* Kursets titel i fuld længde korttitel *(unik, ikke null) - (VARCHAR(45))* Forkortelse af kursets titel udir *(unik, ikke null) - (INT)* Kursus-nummer. Nummeret omtales som udir, men er et nummer som forbliver det samme på tværs af alle versioner af uddannelsesdirektiver for et bestemt kursus.

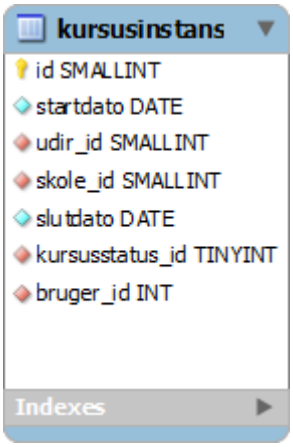
BRUGER_ADMINISTRERER_KURSUS



Primærnøgle(r):	bruger_id *(INT)* Brugerens id kursus_id *(SMALLINT)*

	Id for det kursus som brugeren er kursusansvarlig for.

KURSUSINSTANS

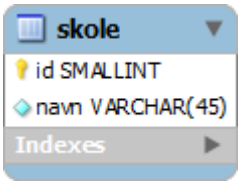


Primærnøgle(r):	id *(SMALLINT)* Autonummereret identifikationsnummer
Fremmednøgle(r):	udir_id *(SMALLINT)* Fremmednøgle til uddannelsesdiretivet. skole_id *(SMALLINT)* Fremmednøgle til den skole på hvilken kurset afholdes. kursusstatus_id *(TINYINT)* Fremmednøgle til kursusstatus. Vi vælger ikke at skrive status direkte på kursusinstansen, fordi det vil duplikere konkrete tilstande ud på rigtig mange rækker i kursusinstans-tabellen. bruger_id *(INT)* Fremmednøgle til den kursusinsvalige som afholder denne specifikke kursusinstans.
Attributter:	startdato *(ikke null) - (DATE)* Dato for kursusstart. slutdato *(ikke null) - (DATE)* Dato for sidste dag på kurset.

KURSUSSTATUS

Primærnøgle(r):	id *(TINYINT)* Autonummereret identifikationsnummer
Attributter:	type *(unik, ikke null) - (VARCHAR(45))* Navnet på kursustilstanden. Fx “Fyldt op”, “Aflyst” og lignende.

SKOLE

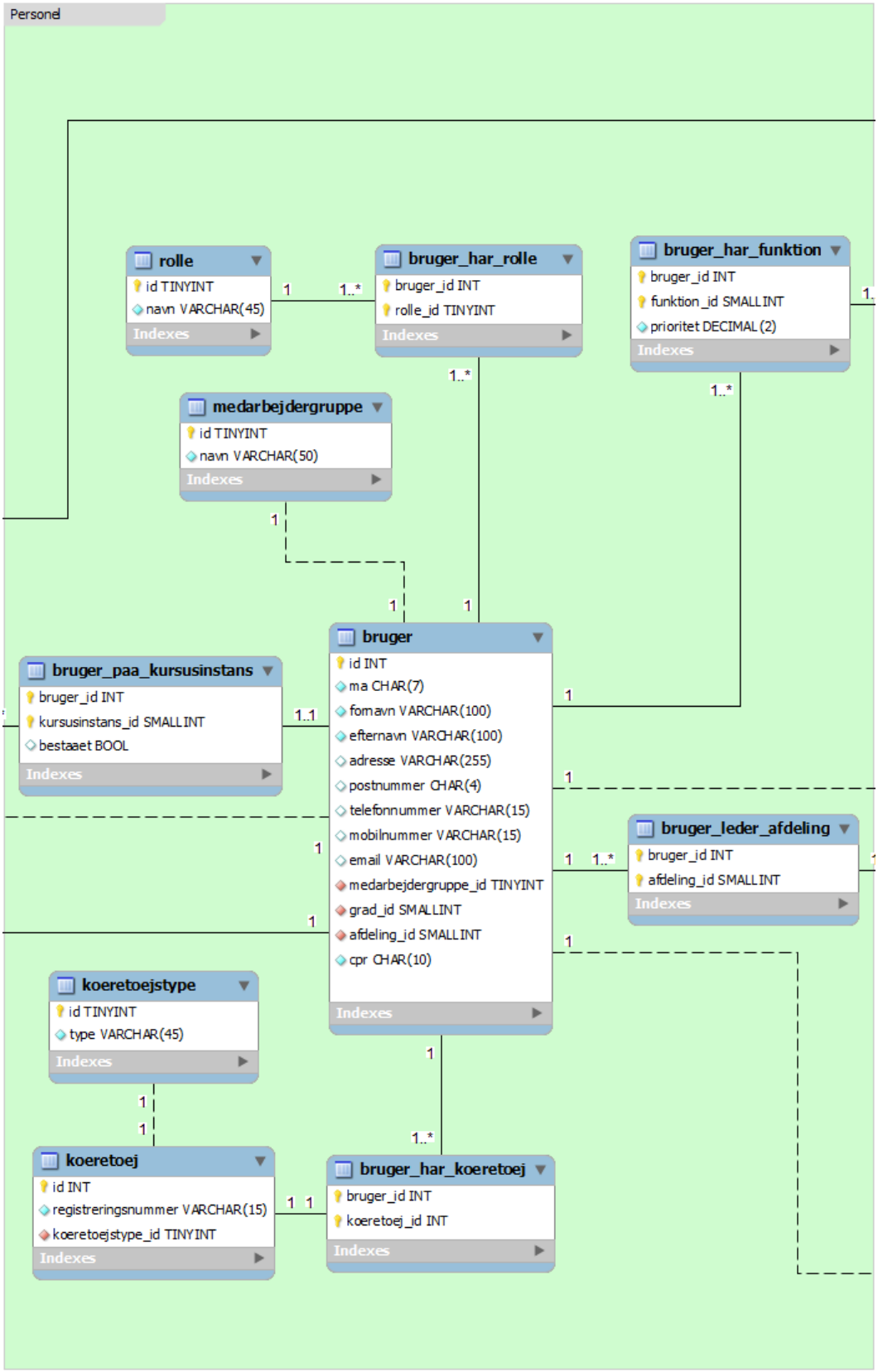


Primærnøgle(r):	id *(SMALLINT)* Autonummereret identifikationsnummer
Attributter:	navn *(unik, ikke null) - (VARCHAR(45))* Navnet på skolen. Fx “Slipshavn”.

PERSONEL

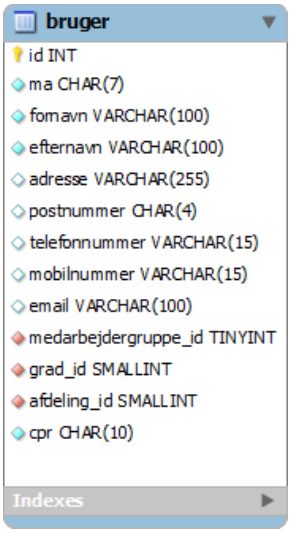
Følgende entiteter udgør personel i databasediagrammet:

- bruger
- rolle
- bruger_har_rolle
- medarbejdergruppe
- bruger_paa_kursusinstans
- bruger_har_koeretoej
- koeretoej
- koeretoejstype
- bruger_leder_afdeling
- bruger_har_funktion



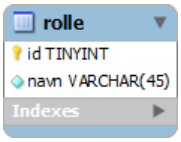
FIGUR 17 - DATABASEDIAGRAM OVER PERSONEL

BRUGER



Primærnøgle(r):	**id** *(INT)* Autonummereret identifikationsnummer
Fremmednøgle(r):	**medarbejdergruppe_id** *(TINYINT)* Fremmednøgle til medarbejdergruppe **grad_id** *(SMALLINT)* Fremmednøgle til brugerens grad. **afdeling_id** *(SMALLINT)* Fremmednøgle til den afdeling som brugeren er tilknyttet.
Attributter:	**ma** *(unik, ikke null) - (CHAR(7))* Brugerens medarbejdernummer **fornavn** *(ikke null) - (VARCHAR(100))* Brugerens fornavn **efternavn** *(ikke null) - (VARCHAR(100))* Brugerens efternavn **adresse** *(VARCHAR(255))* Brugerens adresse **postnummer** *(CHAR(4))* Brugerens postnummer **telefonnummer***(VARCHAR(15))* Brugerens telefonnummer **mobilnummer** *(VARCHAR(15))* Brugerens mobilnummer **email** *(VARCHAR(100))* Brugerens email **cpr** *(unik, ikke null) - (CHAR(10))* Brugerens CPR-nummer

ROLLE

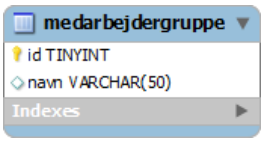


Primærnøgle(r):	**id** *(TINYINT)* Autonummereret identifikationsnummer
Attributter:	**navn** *(unik, ikke null) - (VARCHAR(45))* Navn på rollen.

BRUGER_HAR_ROLLE

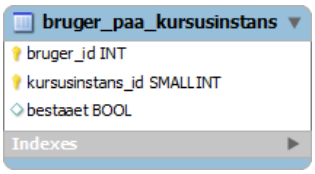
Primærnøgle(r):	**bruger_id** *(INT)* Brugerens id **rolle_id** *(TINYINT)* Rollens id

MEDARBEJDERGRUPPE



Primærnøgle(r):	**id** *(TINYINT)* Autonummereret identifikationsnummer
Attributter:	**navn** *(unik, ikke null) - (VARCHAR(50))* Navn på medarbejdergruppen.

BRUGER_PAA_KURSUSINSTANS



Primærnøgle(r):	**bruger_id** *(INT)* Brugerens id **kursusinstans_id** *(SMALLINT)* Kursusinstansens id
Attributter:	**bestået** *(BOOL)* En boolsk værdi som angiver om medarbejderen har bestået kurset. Attributten kan desuden være null, for at indikere at medarbejderen endnu ikke er blevet vurderet (Fx fordi kurset blev aflyst eller endnu ikke har været afholdt.)

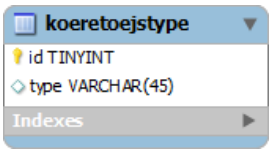
BRUGER_HAR_KORETOEJ

Primærnøgle(r):	**bruger_id** *(INT)* Brugerens id. **koeretoej_id** *(INT)* Køretøjets id.

KOERETOEJ

Primærnøgle(r):	**id** *(INT)* Autonummereret identifikationsnummer
Fremmednøgle(r):	**koeretoejstype_id** *(TINYNT)* Fremmednøgle til køretøjets type.
Attributter:	**registreringsnummer** *(unik, ikke null) - (VARCHAR(15))* Køretøjets registreringsnummer

KOERETOEJSTYPE

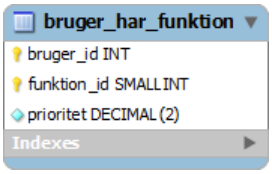


Primærnøgle(r):	**id** *(INT)* Autonummereret identifikationsnummer
Attributter:	**type** *(unik, ikke null) - (VARCHAR(45))* Køretøjets type, fx “personbil” eller “motorcykel”.

BRUGER_LEDER_AFDELING

Primærnøgle(r):	**bruger_id** *(INT)* Id for for brugeren. **afdeling_id** *(SMALLINT)* Id for den afdeling som brugeren er leder for.

BRUGER_HAR_FUNKTION



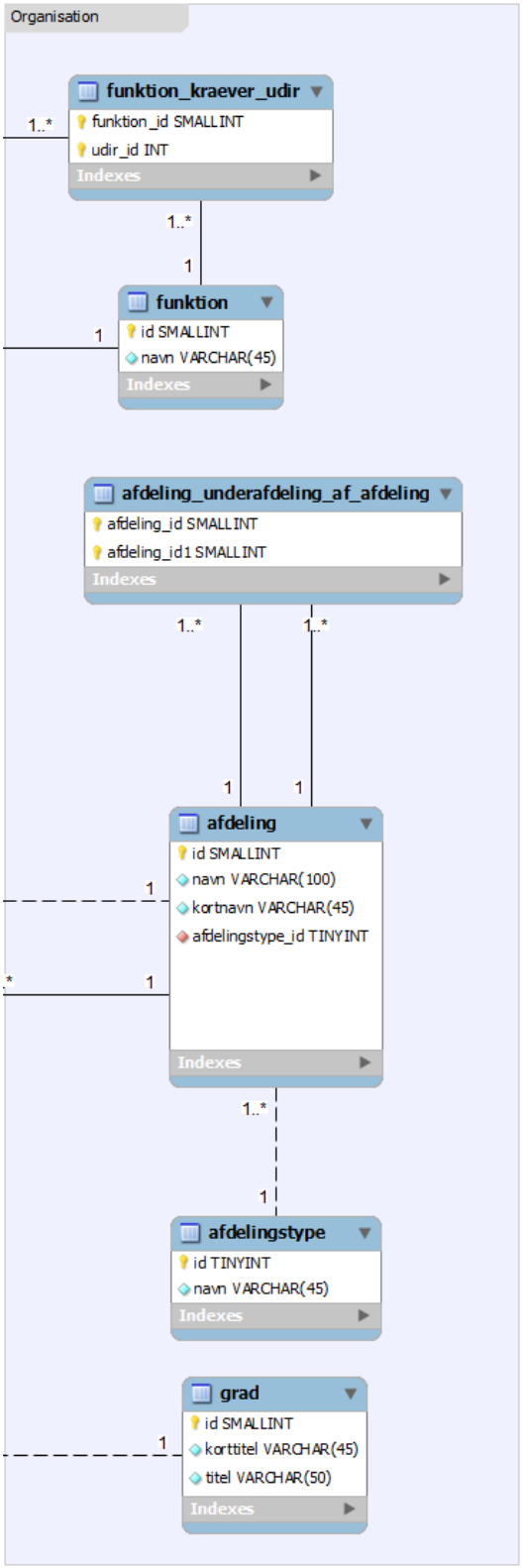
Primærnøgle(r):	**bruger_id** *(INT)* Brugerens id. **funktion_id** *(SMALLINT)* Id for funktionen som brugeren bestrider.
Attributter:	**prioritet** *(ikke null) - (DECIMAL(2))* Funktionens prioritet i forhold til brugeren.

ORGANISATION

Følgende entiteter udgør organisationen:

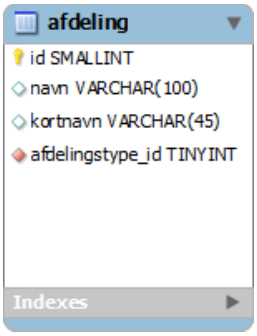
- afdeling
- afdelingstype
- afdeling_underafdeling_af_afdeling
- funktion
- funktion_kraever_udir

- grad



FIGUR 18 - DATABASEDIAGRAM OVER ORGANISATION

AFDELING

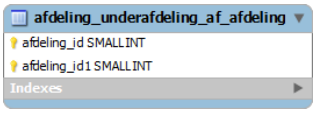


Primærnøgle(r):	**id** *(SMALLINT)* Autonummereret identifikationsnummer
Fremmednøgle(r):	**afdelingstype_id** *(TINYINT)* Fremmednøgle til afdelingstype
Attributter:	**navn** *(unik, ikke null) - (VARCHAR(100))* Afdelingens navn i fuld længde **kortnavn** *(unik, ikke null) - (VARCHAR(45))* Forkortelse af afdelingens navn

AFDELINGSTYPE

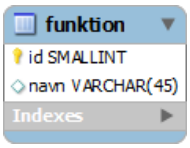
Primærnøgle(r):	**id** *(TINYINT)* Autonummereret identifikationsnummer
Attributter:	**navn** *(unik, ikke null) - (VARCHAR(45))* Navnet på afdelingenstypen. Fx “underafdeling”.

AFDELING_UNDERAFDELING_AF_AFDELING



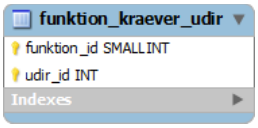
Primærnøgle(r):	**afdeling_id** *(INT)* Id på underafdeling **afdeling_id1** *(INT)* Id på afdeling som underafdeling er underafdeling af

FUNKTION



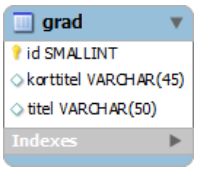
Primærnøgle(r):	**id** *(SMALLINT)* Autonummereret identifikationsnummer
Attributter:	**navn** *(unik, ikke null) - (VARCHAR(45))* Funktionens navn. Fx “Geværskytte”.

FUNKTION_KRAEVER_UDIR



Primærnøgle(r):	**funktion_id** *(SMALLINT)* Id for funktionen som denne relation tager udgangspunkt i **udir_id** *(INT)* Id på uddannelsesdirektivet, der kræves af funktionen

GRAD



Primærnøgle(r):	**id** *(INT)* Autonummereret identifikationsnummer
Attributter:	**korttitel** *(VARCHAR(45))* Forkortelsen for gradens titel **titel** *(unik, ikke null) - (VARCHAR(50))* Gradens titel. Fx “Løjtnant”.

DATATYPER

I gennemgangen af datamodellen benyttes en lang række forskellige datatyper. I dette afsnit argumenteres der for valget af datatyper i de enkelte tilfælde, med henblik på enten at spare plads eller øge ydelsen af systemet.

INT VS SMALL- OG TINYINT

I en række tabeller benyttes unsigned SMALLINT frem for standarden INT til ID-kolonnen. SMALLINT dækker intervallet 0-65535, hvilket er mere end rigeligt til tabeller som f. eks. afdeling, kursus, udir og funktion, da det er yderst usandsynligt at disse tabeller nogensinde vil indeholde mere end 65535 rækker. Enkelte tabeller har endnu færre rækker som f. eks. afdelingstype, koeretoejstype og kursusstatus, hvor der er brugt unsigned TINYINT som dækker intervallet 0-255. Bruger-tabellen benytter som den eneste tabel et ID af typen unsigned INTEGER, der dækker intervallet 0-4294967295, da der i skrivende stund er lidt over 50.000 medlemmer i hjemmeværnet.

Primær- og fremmednøgler er altid unsigned og de er aldrig NULL givet deres funktion som nøgler.

CHAR VS. VARCHAR

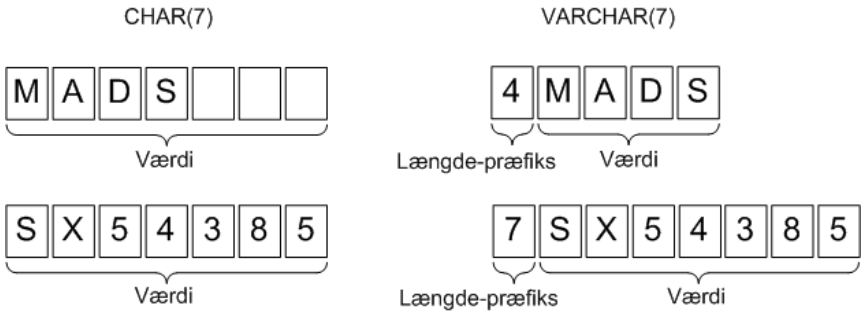
Til tekst-strenge benyttes CHAR(n) når længden er kendt, hvor længden er n. Et eksempel på dette er MA-nummer, der altid er 7 karakterer langt, samt postnummer på 4 karakterer. Til længere tekster, hvor længden ikke er fast men dog stadig kan fastsættes til et maksimum, benyttes VARCHAR(n).

CHAR og VARCHAR minder på mange punkter om hinanden og har følgende egenskaber:

- Den maksimale længde datatypen kan indeholde angives når tabellen oprettes
- Værdien gemmes samme sted som resten af den pågældende række, på den fysiske lagerplads

Forskellen mellem de to ligger i størrelsen feltet optager på den fysiske lagerplads, hvor CHAR optager plads proportionalt med det maksimale antal af karakterer. Registreringsnummer som er et felt af typen CHAR(7) vil da optage 7 BYTE ligegyldigt om værdien er "SX54385" eller "Mads". I modsætning til CHAR vil VARCHAR variere hukommelsesforbruget i forhold til den aktuelle længde af feltet fremfor den maksimalt mulige længde. For at holde styr på hvor på disken det næste felt begynder, starter en VARCHAR med én eller to byte der angiver antallet af byte der skal læses frem. Én byte hvis der maksimalt kan være 255 tegn og to hvis det er flere end 255.

Dette giver følgende billede for strengene "SX54385" og "Mads":



FIGUR 19 - PLADSFORBRUGET FOR CHAR OG VARCHAR

Ud fra ovenstående kan følgende siges om pladsforbruget for de to strenge samt en tom streng:

Værdi	CHAR(7)	Pladsforbrug	VARCHAR(7)	Pladsforbrug
''	' '	7 byte	''	1 byte
'Mads'	'Mads '	7 byte	'Mads'	5 byte
'SX54385'	'SX54835'	7 byte	'SX54385'	8 byte

Ovenstående tabel viser en klar forskel i pladsforbruget og er en hjælp til at afgøre hvornår den ene bruges frem for den anden, da det overhead der introduceres ved VARCHAR er så lille at VARCHAR burde resultere i et lavere pladsforbrug, medmindre størstedelen af rækkerne tilnærmer sig den maksimale længde af kolonnen. Et endegyldigt svar på hvilken af de to der giver det bedste resultat, ville kræver en dybdegående benchmark med store mængder data.

Telefonnumre er sat til en VARCHAR(15) jvf. ITUs E.164-standard (se appendix), som beskriver at internationale telefonnumre maksimalt har 15 cifre. Dette valg er truffet for netop at understøtte internationale telefonnumre, på folk der kan være udstationerede.

TEXT

I modsætning til CHAR og VARCHAR, der gemmes sammen med resten af tabellen, bliver værdien af et TEXT-felt gemt et eksternt sted. Det eneste der gemmes i tabellen er en pointer til adressen hvor den faktiske værdi er gemt, som i stedet er gemt et andet sted på disken.

CONSTRAINTS

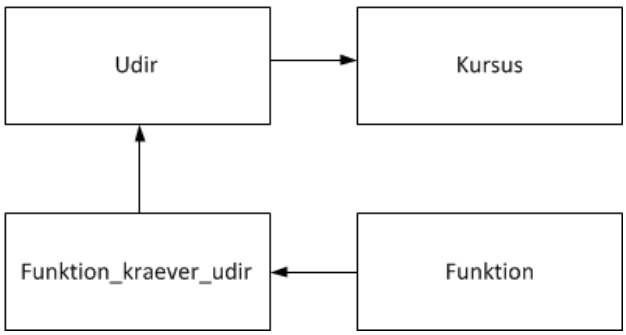
Vi har sikret at der er constraints på alle foreign keys. Dette er gjort for at datamodellen på den måde kan hjælpe applikationslaget med at sikre konsistent data. Det sikrer at applikationslaget ikke ved en fejl tillader at slette en række som der henvises til fra en anden tabel, med inkonsistent data til resultat. Hvis man havde haft en løsning med denne type constraints i datamodellen på hvs-info.dk ville man have undgået en stor del af de inkonsistenser som der bl.a. beskrives Appendix.

SQL-EKSEMPEL

Da fokus for dette projekt har været at definere og implementere fundamentet for det system som vi specificerer, vil vi ikke gå i dybden med applikationsspecifikke detaljer. Vi har dog skrevet en enkelt SQL-forespørgsel som illustrerer hvordan det SQL som skal benytte databasen, skal opbygges. Nedenfor findes et SQL-query som besvarer forespørgslen "Hvilket kursus er forudsat af funktionen KDOBM (Kommandobefalingsmand)?":

```
SELECT udir, korttitel, langtitel
FROM kursus
    JOIN udir ON kursus.udir_id = udir.id
    JOIN funktion_kraever_udir ON udir.id = funktion_kraever_udir.udir_id
    JOIN funktion ON funktion_kraever_udir.funktion_id = funktion.id
WHERE navn = 'KDOBM'
```

Nedenstående diagram illustrerer den konceptuelle vej som SQL-forespørgslen navigerer - Fra funktions-entiteten til kursus-entiteten:



NORMALFORMER

Vi har designet databaseskemaet så det overholder normalformerne op til tredje normalform. Dette er gjort for at opnå et skema som overholder industriens standard for en velstruktureret database - Fordelene ved dette er bl.a. at man undgår duplikeret data og lignende uhensigtsmæssigheder.

I det følgende vil i gennemgå hver enkelt normalform op til og med tredje normalform, hvor vi for hver enkelt normalform vil argumentere for at vores skema overholder normalformen.

FØRSTE NORMALFORM

For at en database overholder første normalform skal den overholde følgende krav:

- Ingen sortering i databasen som er baseret på rækkefølgen af rækker eller kolonner.
- Ingen replikerede rækker.
- Hvert eneste felt indeholder kun præcis én domæneværdi
- Alle kolonner har regulære værdier, det vil sige at der ikke er skulte komponenter

SORTERING AF RÆKKER

Der er ingen steder i vores database hvor vi har sortering baseret på rækkefølge af dataen i databasen. Det eneste sted hvor vi skaber rækkefølge i databasen er i funktion for en bruger. Her svarer rækkefølgen til funktionens prioritet, og denne rækkefølge er afgjort af kolonnen "prioritet", og ikke kolonner eller rækkers rækkefølge i skemaet.

REPLIKEREDE RÆKKER

Et oplagt sted hvor man som database-designer kunne være fristet til at bruge replikerede rækker, er netop for en brugers funktion. Her har den nuværende applikation tre funktioner for en bruger, nemlig "primær funktion", "sekundær funktion" samt "tertiær funktion". Det kunne være oplagt at lave netop tre fremmednøgler fra bruger til funktion, men dette vil bryde med den første normalform. I stedet kræver denne at man opretter en mange-til-mange-relation mellem bruger og funktion, og så skal applikationen sikre at der maksimalt oprettes tre funktioner per bruger, og sikre at prioriteterne er korrekte. Vi har undgået replikerede rækker i vores databasedesign.

FLERE VÆRDIER I ET FELT

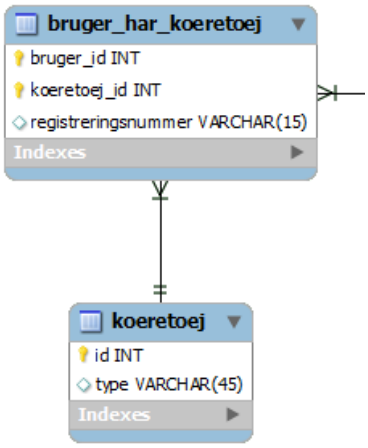
Her kunne man fx have valgt at lægge både mobil-nummer og telefonnummer i samme felt og så udnævne et bestemt tegn, så som mellemrum, som telefonnummer-separator. Dette ville være i strid med første normalform, så derfor har vi undgået denne type løsninger.

SKJULTE VÆRDIER

Vi har undgået alle former for skjulte værdier i vores skema.

ANDEN NORMALFORM

Hvis et databaseskema skal være i anden normalform skal det være i første normalform, og for enhver tabel skal skal alle attributter afhænge af hele nøglen for tabellen. Det vil sige at hvis man har en primærnøgle som består af flere attributter så skal alle øvrige attributter afhænge af hele primærnøglen.



Ser man eksempelvis på vores tabel "bruger_har_koeretoej" til højre: Her har man brug for at repræsentere tre informationer bruger_id, køretøjtypen samt registreringsnummer. En oplagt løsning vil være nedenstående, som ville spare en tabel i forhold til den løsning vi har valgt:

Ovenstående design ville fejle anden normalform fordi bruger_id og koeretoej_id som tilsammen udgør primærnøglen, men registreringsnummer afhænger kun af koeretoej_id.

TREDJE NORMALFORM

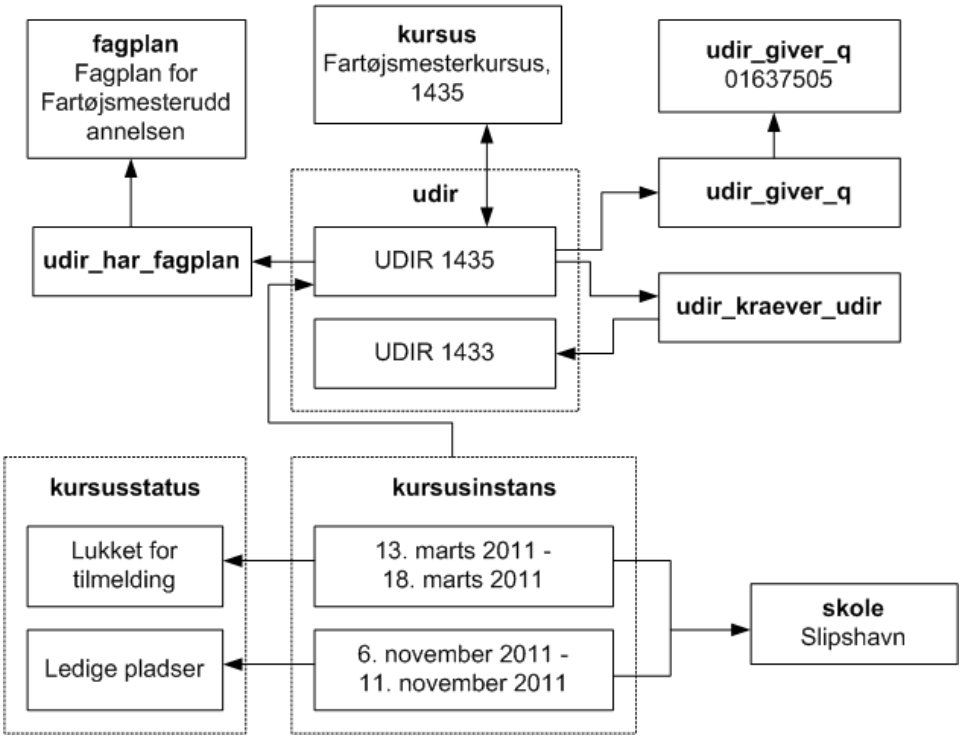
Et databaseskema er i tredje normalform hvis det er i anden normalform og alle attributter som ikke tilhører en kandiat-nøgle, og er direkte afhængig af enhver kandidat-nøgle.

Tog man og repræsenterede både oplysninger om køretøj-type og registreringsnummer på bruger-tabellen ville man bryde med med tredje normal-form. Dette skyldes at køretøj-typen ville afhænge af registreringsnummeret, som herefter ville afhænge af brugeren. Køretøjstype ville altså transitivt afhænge af brugeren, i stedet for at afhænge direkte. Denne type løsninger har vi undgået.

EKSEMPLER

I det følgende afsnit vil gennemgå konkrete brugere og kurser, for at vise hvordan de ville passe ind i datamodellen.

I det følgende har vi taget udgangspunkt i kurset "Fartøjsmesterkursus, 1435". Vi har medtaget de relationer som beskriver det specifikke kursus, sådan som det så ud d. 6. juni 2011.

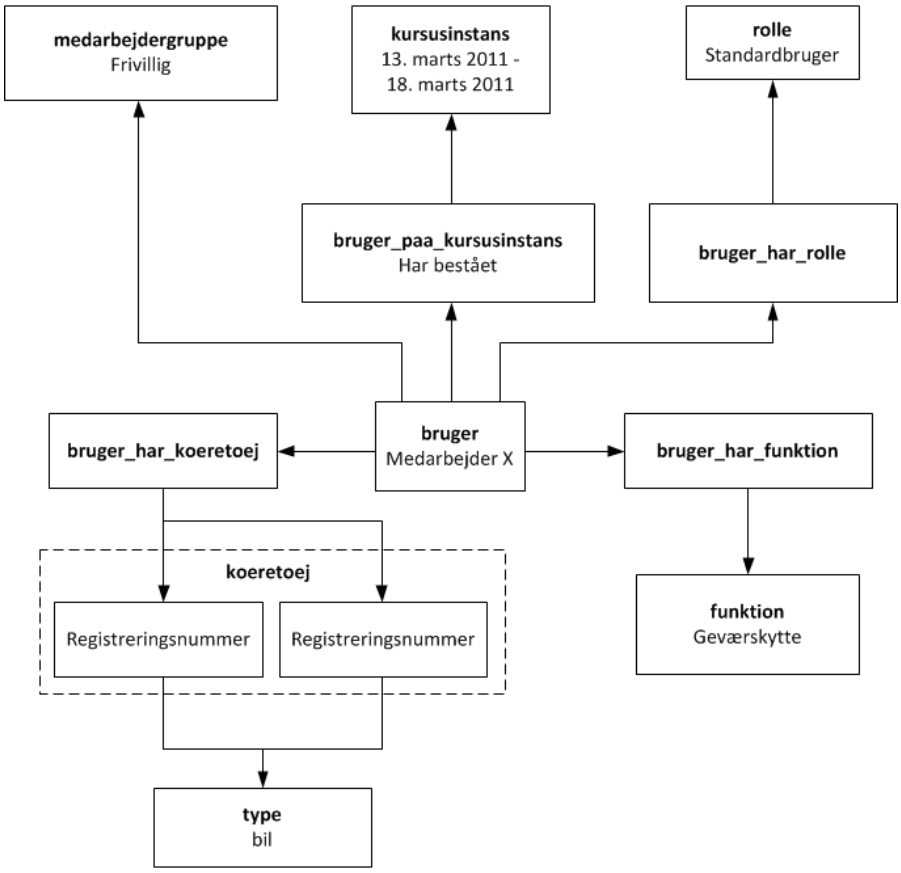


FIGUR 20 - 'FARTØJSMESTERKURSUS, 1435'

kursus	Kurset Fartøjsmesterkursus har det korte navn "FARM". Det har Udir nummer 1435, og henviser til sit aktuelle uddannelsesdirektiv
udir	Det aktuelle uddannelsesdirektiv beskriver detaljer om uddanelsen, så som "mål" og "formål" for kurset. I dette tilfælde er formålet: "Kursus skal give eleven det nødvendige grundlag for at bestride funktionen som fartøjsmester i en flotille."
udir_kraever_udir	Den aktuelle version af uddanelsesdirektivet UDIR 1435 forudsætter at kursusdeltagere har gennemført den aktuelle version af kurset UDIR 1433 (Motorpasserkursus).
udir_giver_q	Da et kursus kan give et vilkårligt antal Q benyttes en relationstabel til henvise til de Q-numre som kurset giver.
q	Fartøjsmesterkursus giver ét Q-nummer, nemlig 01637505.

udir_har_fagplan	Denne relationstabel er indskudt mellem udir og fagplan for at give mulighed for at et uddannelsesdirektive kan have et vilkårligt antal fagplaner.
fagplan	Kurset har én fagplan. Denne fagplan er oprettet i December 2010, og indeholder informationer om Fartøjsmesterkursus.
kursusinstans	Der vises for kurset de instanser af kurset som tidsmæssigt ligger relativt tæt på den aktuelle dato. I skrivende stund er dette ét afholdt kursus, og ét kursus som afholdes ude i fremtiden.
kursusstatus	En tabel benyttes til at indeholde alle kursusstatus'er. Dette skyldes at der er et endeligt antal status'er, og i stedet for at repræsentere de samme status'er for en lang række kursusinstanser, er det mere effektivt at have fremmednøgler fra hver enkelt kursusinstans til den aktuelle kursusstatus.
skole	Begge instanser af det aktuelle kursus afholdes på skolen Slipshavn.

Vi illustrerer nu en fiktiv bruger som har deltaget på Fartøjsmesterkursus:



FIGUR 21 - BRUGER PÅ FARTØJSMESTERKURSUS

Bruger	Den fiktive medarbejder X repræsenteres i tabellen bruger.
Medarbejdergruppe	Den fiktive medarbejder X tilhører medarbejdergruppen “Frivillige”.
Bruger_kursusinstans	Denne relationstabel er indskudt mellem bruger og kursusinstans for at give mulighed for at en bruger kan have deltaget på et vilkårligt antal kurser.
Kursusinstans	Der relateres til kursusinstansen for kurset “Fartøjsmester” fra det foregående diagram. En bruger vil have en relation til samtlige kursusinstanser som vedkommende har deltaget på.
Bruger_har_rolle	Relationstabel for at give mulighed for et vilkårligt antal roller.
Rolle	Medarbejder X har kun én rolle, nemlig *standardbruger*.
Bruger_har_koeretoej	Relationstabel for at give mulighed for et vilkårligt antal køretøjer.
Koeretoej	I eksemplet har Medarbejder X to køretøjer.
Type	Begge køretøjer har typen “bil”.
Bruger_har_funktion	Relationstabel for at give mulighed for et vilkårligt antal funktioner.
Funktion	Medarbejder X har kun én funktion, nemlig funktionen *geværskytte*.

DISKUSSION

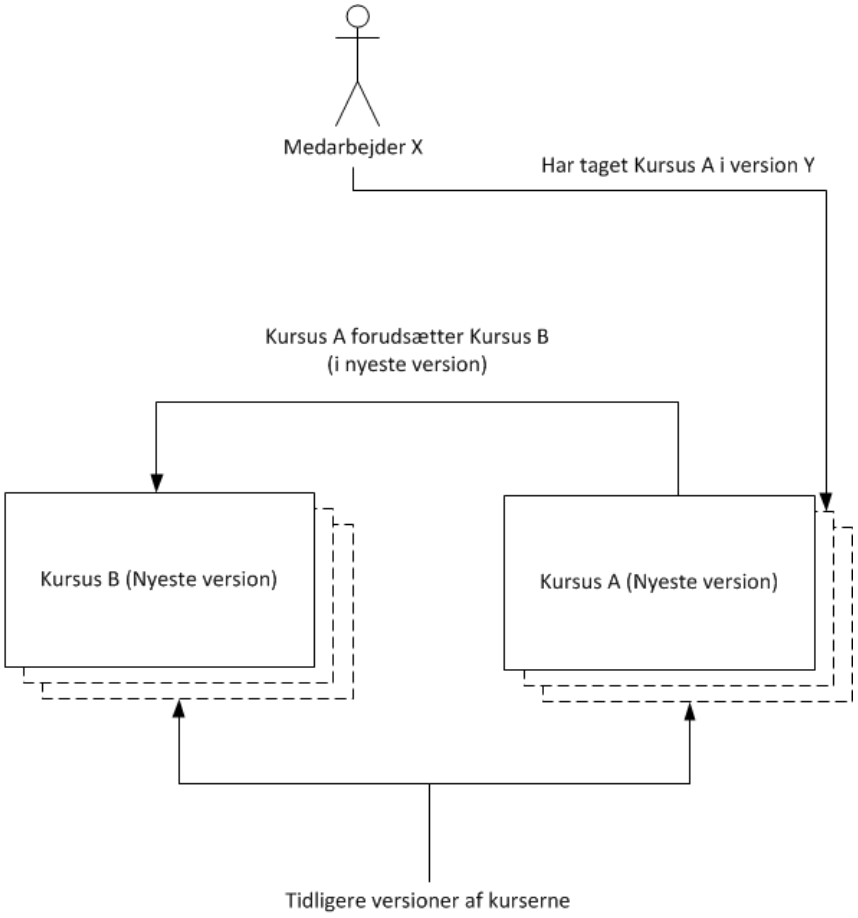
Vores løsning til problemet består af en kravspecifikation og en datamodel. Datamodellen er til dels en implementation af kravspecifikationen, fordi den er opbygget på en måde så det bliver muligt at udføre alle use-cases, samtidig med at den giver mulighed for at indeholde de entiteter som kravspecifikationen specificerer. Ønsker man at implementere CCMS vil kravspecifikation og datamodel tilsammen udgøre fundamentet, hvad enten implementationsopgaven blev udliciteret, eller vi selv løste den. I dette kapitel vil vi forsøge at distancere os fra resultatet og foretage en vurdering af vores løsning, dels i forhold til hvad vi anser for at være en optimal løsning, og dels i forhold til de krav fra kunden, som vi har været underlagt.

Både vi og hjemmeværnet mener i sagens natur at en implementation af det system vi har specificeret vil være langt bedre for brugerne end den nuværende løsning. Der er dog også nogle steder hvor vi mener at man kunne skære igennem og få en bedre løsning, hvis man undlod at repræsentere entiteterne på samme måde som man gjorde i det nuværende system. I det omfang det har været muligt, har vi påpeget det, når vi har stødt på denne type uhensigtsmæssigheder.

KURSER OG UDDANNELSESDIREKTIVER

Den måde som vores datamodel repræsenterer kurser og uddannelsesdirektiver er gennemgået i detaljer i tidligere afsnit, og er forholdsvis kompliceret, fordi der skal være understøttelse for tidligere versioner af uddannelsesdirektiver.

Her havde vi foretrukket en løsning hvor man droppede idéen om at opsplitte konceptet “et kursus” i entiteterne *kursus* og *uddannelsesdirektiv*, hvor uddannelsesdirektiver findes i et antal versioner for et kursus. Vi mener at man ville få en langt mere gennemskuelig løsning hvis man nøjedes én entitet *kursus* som havde et versionsnummer. Når et kursus så blev redigeret i en sådan grad, at man med den nuværende løsning skulle oprette et nyt uddannelsesdirektiv, ville man i stedet opdatere kursets versionsnummer. Da man ønsker at beholde historiske informationer om ændrede kurser, ville en opdatering af et versionsnummer i praksis betyde at man opretter et nyt kursus med indeholdet af det tidligere kursus, og det nyoprettede får et versionsnummer som er 1 højere end versionsnummeret for det tidligere kursus. Alle kurser som forudsætter andre kurser vil per definition altid forudsætte den nyeste version af kurset de forudsætter.



FIGUR 22 - MEDARBEJDER HAR TAGET KURSUS A I VERSION Y

Ved samme lejlighed kunne man ændre det således at man undlod at benytte et Q-nummer i den nuværende form til at identificere at man har gennemført et kursus i en bestemt version. Da et Q-nummer er et arbitrært tal, er det de færreste som umiddelbart vil kunne gennemskue hvilket kursus det vedrører. I stedet kunne man bruge kursus-nummeret (Det som i vores design hedder udir) sammen med versionsnummeret, sådan at man fx kunne sige at man havde gennemført 1201-001 som ville beskrive at have gennemført 1201 (Politifunktionskursus Særlig Hjælp) i version 001.

Vi mener også at det ville være simplere for alle, hvis man droppede idéen om at man tog bestemte "Q", og i stedet bare omtalte hvilke kurser man havde gennemført og eventuelt i hvilken version, så som med "1201-001"-eksemplet. På den måde vil man også komme udover kompleksiteten ved at kurser kan give nul, et eller flere Q-numre. Vi tror at hvis man brugte et system som havde fuldstændig styr på hvilke kurser en medarbejder har gennemført, og som indeholdt information om hvilke kurser der forudsatte hvad, vil brugerne hurtigt tillægge konceptet "Q" mindre betydning, fordi man kan stole på at systemet har denne viden.

Vi har dog arbejdet under nogle krav som dikterede at vi repræsenterede uddannelser på samme måde som det nuværende system, og derfor havde vi ikke mulighed for at lave ovenstående løsning.

Der er dog også en lang række steder hvor vi har gennemtrumfet løsninger som vi mener er langt bedre end det som der blev lagt op til. Dette er eksempelvis i vores datamodel, hvor vi i en lang række tilfælde har opbygget vores databaseskema så det ikke er bundet af arbitrære antal gentagelser af koncepter, så som at en medarbejder

kunne have præcis tre køretøjer og tre funktioner. Her har vi designet en datamodel som kan indeholde et vilkårligt antal køretøjer og funktioner, herunder også hvor funktionerne har en indbyrdes prioritet, således at datamodellen overholder normalformerne og giver mere fleksibilitet fordi den overfører antals-begrænsningen til applikationen.

Et andet eksempel hvor vi har optimeret ved at stramme op i forhold til de krav som vi har modtaget, er i forbindelse med uddannelsesdirektiver og fagplaner. Her har vi, på baggrund af undersøgelser af deres nuværende dokumenter, designet en datamodel som kan indeholde henholdsvis fagplaner og uddannelsesdirektiver på en mere struktureret måde, end bare ét stort tekstfelt, sådan som det blev efterspurgt i første omgang.

KONKLUSION

Resultatet af dette projekt er at vi med succes har udfærdiget en komplet kravspecifikation samt en datamodel med de funktioner som kravspecifikationen definerer. Til formen for vores kravspecifikation har vi valgt at benytte IEEE's industristandard for specifikation af softwaresystemer, fordi vi på den måde har sikret at et softwarefirma som eventuelt skal implementere kravspecifikationen vil have optimale forudsætninger. Under udfærdigelsen af kravspecifikationen har vi begrænset antallet af use cases så vi udelukkende inkluderer de centrale, men den datamodel vi har udviklet er fleksibel så systemet kan udvides med flere use cases efterfølgende.

Arbejdsopgaven har været udfordrende og til tider vanskelig, både fordi den har involveret konkakt med en ekstern tredjepart, men også fordi opgaven til at starte med var meget løst defineret, og vi skulle bruge lang tid på at anlysere, isolere og afgrænse det problem vi skulle løse. Vi måtte erkende at for at kunne designe systemet så vi var sikre på at det dækker alle krav, også dem som vi ikke eksplicit fik oplyst, var det nødvendigt at foretage en grundig analyse af Hjemmeværnets nuværende systemer, og hvordan de benyttes.

I den forbindelse har vi også investeret en del tid på at finde system i Hjemmeværnets nuværende uddannelsesdatabase, sådan at vi kunne designe en datamodel som var optimal i forhold til indholdet. I den forbindelse har det bl.a. været udfordrende at meget af dataen i uddannelsesdatabasen er inkonsistent, hvilket forværres af at selve sammenhængen mellem uddannelser er forholdsvis kompliceret. Resultatet af vores analyse og kravspecifikation var at vi havde tilstrækkeligt med viden om systemet til at vi kunne udvikle en SQL-baseret datamodel som kan udgøre fundamentet for en implementation af systemet.

Efter at have afsluttet projektet mener vi at man visse steder kunne have truffet nogle valg som kunne have gjort systemet nemmere at arbejde med og forstå, hvis man havde simplificeret nogle af de centrale koncepter, herunder særligt kurser og uddannelsesdirektiver. Dette lå dog udenfor den opgave vi havde fået stillet, og vi mener at det system vi har designet til fulde vil løse det problem som det er designet til.

APPENDIX

PUNKTER I UDDANNELSDIREKTIVER

For at kunne afgøre hvilke felter der skulle være i et uddannelsesdirektiv benyttede vi en række Unix kommandolinje-værktøjer til at finde ud af hvilke overskrifter der fandtes i de uddannelsesdirektiver som findes på http://hvs-info.dk. Disse tal satte os i stand til at træffe kvalificerede valg i forhold til at designe en optimal datamodel for uddannelsesdirektiver.

Vi nåede frem til tallene på følgende måde:

Først hentes alle uddannelsesdirektiver:

```
for i in $(seq 1000);
    do wget "http://hvs-info.dk/uddannelser.asp?mode=info&ud_ID=$i";
done
```

Vi havde forinden afgjort at serveren benyttede ID'er for uddannelsesdirektiver i intervallet 1-1000. Her efter benyttedes kommandoen ls | wc -l til tælle hvor mange uddannelsesdirektiver der fandtes. Det præcise antal er i skrivende stund: 436.

Vi kunne herefter afgøre at alle overskrifter i uddannelsesdirektiverne tilsyneladende var omkranset af et HTML -tag. Denne information kunne benyttes til at isolere netop overskrifterne ved hjælp af følgende kommando:

```
grep -oh '<strong>[^<]*' * | cut -d\> -f 2 | sort | grep -v "^$" | uniq -c | sort -n
```

Ved hjælp af denne kommando hentes først alle strenge som starter med og som slutter med < (ikke inkluderet). Herefter hentes alt efter >. Herefter sorteres linjerne og alle tomme linjer fjernes. Antallet af forskellige overskrifter tælles, og der sorteres efter største antal. Resultatet findes nedenfor. I resultatet nævnes kun de overskrifter som indgår i mindst tre uddannelsesdirektiver, og der er håndteret HTML entities og et par andre mindre oprydninger.

```
  3 INDHOLD
  3 Kontrol
  3 Kurslængd
  3 MHV
  3 Optagelsesbetingelser
  3 Samlet varighed
  3 Slutmål
  4 FORMåL
  4 Forudsætninger
  4 HHV og FHV
  4 Innhold
  4 Krav
```

```
  4 Målgruppe
  4 Tilstedemoduler
  5 Brevskolen udsendes i forbindelse med optagelse på tilstedeværelseskursus
  5 Utdanningsmål
  5 Videreuddannelse
  6 Bemærk
  6 Optagelsesbetingelser
  6 Situation
  7 Generelt
  7 Omfang
  8 Bemærkninger
 11 Forudsætning for optagelse
 12 Beskrivelse
 12 Samlet varighed
 12 Videre uddannelse
 13 Bemærkning
 17 Direktivet omfatter følgende fag
 19 Omfang
 28 Kursusbeskrivelse
 30 Indhold
 32 Studiekreds
 38 Tid
 40 Varighed
107 Mål
146 Indhold
221 Formål
```

ITUS STANDARDER FOR TELEFONNUMRE

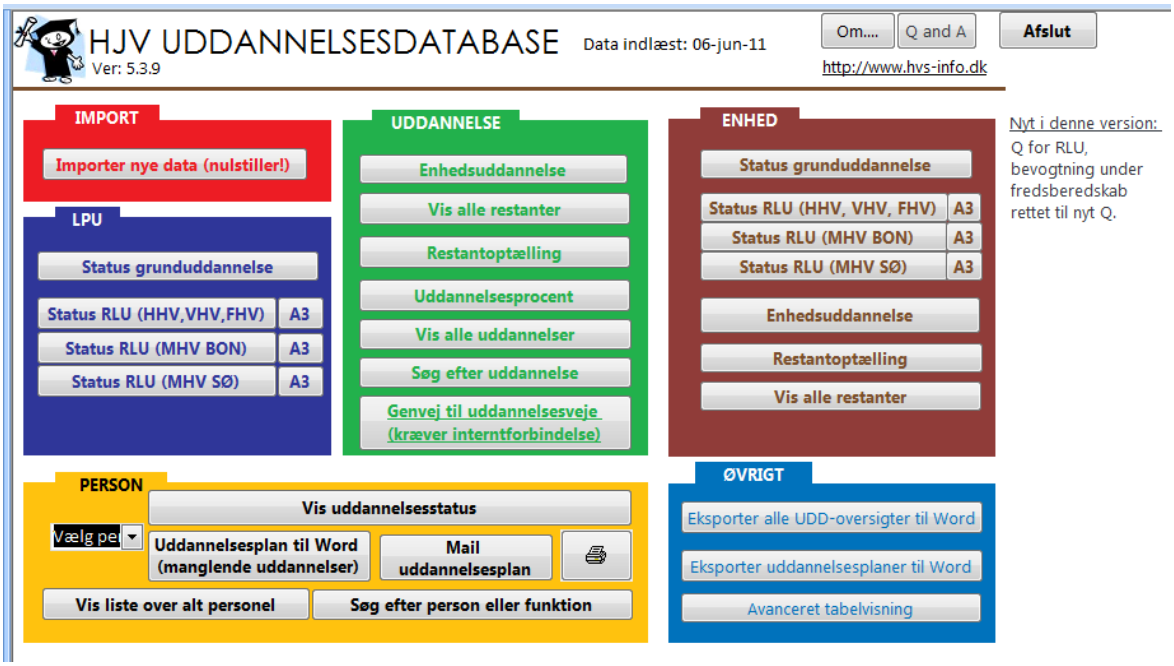
ITU (International Telecommunication Union) har udarbejdet E.164-standarden, med navnet "The international public telecommunication numbering plan", der kort sagt siger at et international telefonnummer maksimalt må være 15 cifre.

The international public telecommunication numbering plan:

http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=E.164

HJEMMEVÆRNETS ACCESSDATABASE TIL REGISTRERING AF KURSUSINFORMATION

Nedenstående er forsiden for Hjemmeværnets uddannelsesdatabase. Forsiden indeholder ikke fortrolig information og er derfor ikke uredigeret.



BRUGERINFORMATIONSSIDEN PÅ [HJV.DK](HJV.DK)

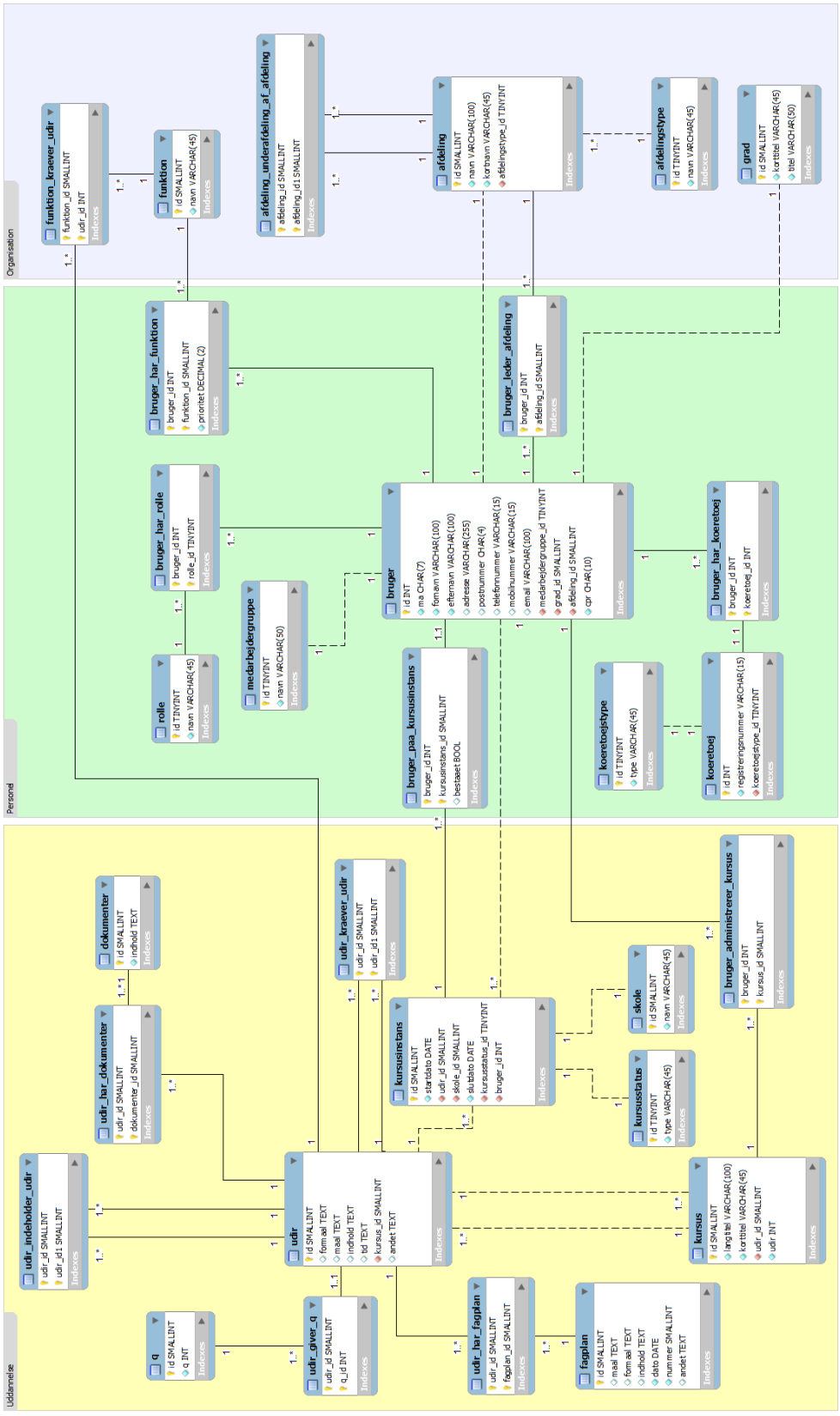
Bemærk at nedenstående side har fået fjernet personfølsomme informationer.

FEJL OG INKONSISTENSER I HVS-INFO.DK

Nedenfor findes nogle få eksempler på inkonsistenser som vi har observeret under vores forsøg på at få et overblik over uddannelserne:

1. På oversigtssiden vises der for kurset 4010 Skydelærerkursus at denne uddannelse giver 10 forskellige Q'er. Men går man ind i uddanelsesdirektivet for denne uddannelse ses det at uddannelsen i virkeligheden giver 13 forskellige Q'er.
2. I uddannelsesdirektivet for 4000 Grundlæggende skydelærerkursus henvises der til at 2002 Alment befalingsmandskursus er en forudsætning. Dette kursus eksisterer dog ikke, men det gør i stedet 2000 Alment befalingsmandskursus.
3. I fagplanen for 1038 RLU, Bevogtning under fredsberedskab til RLU henvises der til at der gives Q 02115681 for den indeholdte uddannelse 1041 Almindelig hjælp til polititet. Men denne uddannelse eksisterer ikke, i stedet findes 1200 Politifunktionskursus Almindelig Hjælp. Uddannelsen med Udir 1041 er RLU, Førstehjælp på kamppladsen.

SAMLET DATABASEDIAGRAM



SQL CREATE-SCRIPT

Nedenfor findes det komplette create-script til datamodellen

```
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 `hjv` DEFAULT CHARACTER SET latin1 COLLATE
latin1_swedish_ci ;
USE `hjv` ;

-- -----------------------------------------------------
-- Table `hjv`.`medarbejdergruppe`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`medarbejdergruppe` (
  `id` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `navn` VARCHAR(50) NOT NULL ,
  PRIMARY KEY (`id`) ,
  UNIQUE INDEX `navn_UNIQUE` (`navn` ASC) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`grad`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`grad` (
  `id` SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `korttitel` VARCHAR(45) NOT NULL ,
  `titel` VARCHAR(50) NOT NULL ,
  PRIMARY KEY (`id`) ,
  UNIQUE INDEX `titel_UNIQUE` (`titel` ASC) ,
  UNIQUE INDEX `korttitel_UNIQUE` (`korttitel` ASC) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`afdelingstype`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`afdelingstype` (
  `id` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `navn` VARCHAR(45) NOT NULL ,
  PRIMARY KEY (`id`) ,
  UNIQUE INDEX `navn_UNIQUE` (`navn` ASC) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`afdeling`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`afdeling` (
  `id` SMALLINT NOT NULL AUTO_INCREMENT ,
  `navn` VARCHAR(100) NOT NULL ,
  `kortnavn` VARCHAR(45) NOT NULL ,
  `afdelingstype_id` TINYINT UNSIGNED NOT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_afdeling_type1` (`afdelingstype_id` ASC) ,
  UNIQUE INDEX `kortnavn_UNIQUE` (`kortnavn` ASC) ,
  UNIQUE INDEX `navn_UNIQUE` (`navn` ASC) ,
  CONSTRAINT `fk_afdeling_type1`
    FOREIGN KEY (`afdelingstype_id` )
```

```
    REFERENCES `hjv`.`afdelingstype` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`bruger`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`bruger` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `ma` CHAR(7) NOT NULL ,
  `fornavn` VARCHAR(100) NOT NULL ,
  `efternavn` VARCHAR(100) NOT NULL ,
  `adresse` VARCHAR(255) NULL ,
  `postnummer` CHAR(4) NULL ,
  `telefonnummer` VARCHAR(15) NULL ,
  `mobilnummer` VARCHAR(15) NULL ,
  `email` VARCHAR(100) NULL ,
  `medarbejdergruppe_id` TINYINT UNSIGNED NOT NULL ,
  `grad_id` SMALLINT UNSIGNED NOT NULL ,
  `afdeling_id` SMALLINT UNSIGNED NOT NULL ,
  `cpr` CHAR(10) NOT NULL ,
  PRIMARY KEY (`id`) ,
  UNIQUE INDEX `ma_UNIQUE` (`ma` ASC) ,
  INDEX `fk_bruger_medarbejdergruppe1` (`medarbejdergruppe_id` ASC) ,
  INDEX `fk_bruger_grad1` (`grad_id` ASC) ,
  INDEX `fk_bruger_afdeling1` (`afdeling_id` ASC) ,
  UNIQUE INDEX `cpr_UNIQUE` (`cpr` ASC) ,
  CONSTRAINT `fk_bruger_medarbejdergruppe1`
    FOREIGN KEY (`medarbejdergruppe_id` )
    REFERENCES `hjv`.`medarbejdergruppe` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_bruger_grad1`
    FOREIGN KEY (`grad_id` )
    REFERENCES `hjv`.`grad` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_bruger_afdeling1`
    FOREIGN KEY (`afdeling_id` )
    REFERENCES `hjv`.`afdeling` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`udir`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`udir` (
  `id` SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `formaal` TEXT NULL ,
  `maal` TEXT NULL ,
  `indhold` TEXT NULL ,
  `tid` TEXT NULL ,
  `kursus_id` SMALLINT UNSIGNED NOT NULL ,
  `andet` TEXT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_udir_kursus1` (`kursus_id` ASC) ,
  CONSTRAINT `fk_udir_kursus1`
    FOREIGN KEY (`kursus_id` )
    REFERENCES `hjv`.`kursus` (`id` )
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`kursus`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`kursus` (
  `id` SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `langtitel` VARCHAR(100) NOT NULL ,
  `korttitel` VARCHAR(45) NOT NULL ,
  `udir_id` SMALLINT UNSIGNED NOT NULL ,
  `udir` INT NOT NULL ,
  PRIMARY KEY (`id`) ,
  UNIQUE INDEX `langtitel_UNIQUE` (`langtitel` ASC) ,
  UNIQUE INDEX `korttitel_UNIQUE` (`korttitel` ASC) ,
  INDEX `fk_kursus_udir1` (`udir_id` ASC) ,
  UNIQUE INDEX `udir_UNIQUE` (`udir` ASC) ,
  CONSTRAINT `fk_kursus_udir1`
    FOREIGN KEY (`udir_id` )
    REFERENCES `hjv`.`udir` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`skole`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`skole` (
  `id` SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `navn` VARCHAR(45) NOT NULL ,
  PRIMARY KEY (`id`) ,
  UNIQUE INDEX `navn_UNIQUE` (`navn` ASC) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`kursusstatus`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`kursusstatus` (
  `id` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `type` VARCHAR(45) NOT NULL ,
  PRIMARY KEY (`id`) ,
  UNIQUE INDEX `type_UNIQUE` (`type` ASC) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`kursusinstans`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`kursusinstans` (
  `id` SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `startdato` DATE NOT NULL ,
  `udir_id` SMALLINT UNSIGNED NOT NULL ,
  `skole_id` SMALLINT UNSIGNED NOT NULL ,
  `slutdato` DATE NOT NULL ,
  `kursusstatus_id` TINYINT UNSIGNED NOT NULL ,
  `bruger_id` INT UNSIGNED NOT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_kursusinstans_udir1` (`udir_id` ASC) ,
  INDEX `fk_kursusinstans_skole1` (`skole_id` ASC) ,
```

```
  INDEX `fk_kursusinstans_Kursusstatus1` (`kursusstatus_id` ASC) ,
  INDEX `fk_kursusinstans_bruger1` (`bruger_id` ASC) ,
  CONSTRAINT `fk_kursusinstans_udir1`
    FOREIGN KEY (`udir_id` )
    REFERENCES `hjv`.`udir` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_kursusinstans_skole1`
    FOREIGN KEY (`skole_id` )
    REFERENCES `hjv`.`skole` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_kursusinstans_Kursusstatus1`
    FOREIGN KEY (`kursusstatus_id` )
    REFERENCES `hjv`.`kursusstatus` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_kursusinstans_bruger1`
    FOREIGN KEY (`bruger_id` )
    REFERENCES `hjv`.`bruger` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`bruger_paa_kursusinstans`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`bruger_paa_kursusinstans` (
  `bruger_id` INT UNSIGNED NOT NULL ,
  `kursusinstans_id` SMALLINT UNSIGNED NOT NULL ,
  `bestaaet` TINYINT(1)  NULL ,
  PRIMARY KEY (`bruger_id`, `kursusinstans_id`) ,
  INDEX `fk_bruger_has_kursusinstans_kursusinstans1` (`kursusinstans_id` ASC) ,
  INDEX `fk_bruger_has_kursusinstans_bruger1` (`bruger_id` ASC) ,
  CONSTRAINT `fk_bruger_has_kursusinstans_bruger1`
    FOREIGN KEY (`bruger_id` )
    REFERENCES `hjv`.`bruger` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_bruger_has_kursusinstans_kursusinstans1`
    FOREIGN KEY (`kursusinstans_id` )
    REFERENCES `hjv`.`kursusinstans` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`koeretoejstype`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`koeretoejstype` (
  `id` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `type` VARCHAR(45) NOT NULL ,
  PRIMARY KEY (`id`) ,
  UNIQUE INDEX `type_UNIQUE` (`type` ASC) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`koeretoej`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`koeretoej` (
```

```
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `registreringsnummer` VARCHAR(15) NOT NULL ,
  `koeretoejstype_id` TINYINT UNSIGNED NOT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_koeretoej_koeretoejstype1` (`koeretoejstype_id` ASC) ,
  UNIQUE INDEX `registreringsnummer_UNIQUE` (`registreringsnummer` ASC) ,
  CONSTRAINT `fk_koeretoej_koeretoejstype1`
    FOREIGN KEY (`koeretoejstype_id` )
    REFERENCES `hjv`.`koeretoejstype` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`bruger_har_koeretoej`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`bruger_har_koeretoej` (
  `bruger_id` INT UNSIGNED NOT NULL ,
  `koeretoej_id` INT UNSIGNED NOT NULL ,
  PRIMARY KEY (`bruger_id`, `koeretoej_id`) ,
  INDEX `fk_bruger_has_koeretoej_bruger1` (`bruger_id` ASC) ,
  INDEX `fk_bruger_har_koeretoej_koeretoej1` (`koeretoej_id` ASC) ,
  CONSTRAINT `fk_bruger_has_koeretoej_bruger1`
    FOREIGN KEY (`bruger_id` )
    REFERENCES `hjv`.`bruger` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_bruger_har_koeretoej_koeretoej1`
    FOREIGN KEY (`koeretoej_id` )
    REFERENCES `hjv`.`koeretoej` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`funktion`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`funktion` (
  `id` SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `navn` VARCHAR(45) NOT NULL ,
  PRIMARY KEY (`id`) ,
  UNIQUE INDEX `navn_UNIQUE` (`navn` ASC) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`funktion_kraever_udir`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`funktion_kraever_udir` (
  `funktion_id` SMALLINT UNSIGNED NOT NULL ,
  `udir_id` INT UNSIGNED NOT NULL ,
  PRIMARY KEY (`funktion_id`, `udir_id`) ,
  INDEX `fk_funktion_has_udir_udir1` (`udir_id` ASC) ,
  INDEX `fk_funktion_has_udir_funktion1` (`funktion_id` ASC) ,
  CONSTRAINT `fk_funktion_has_udir_funktion1`
    FOREIGN KEY (`funktion_id` )
    REFERENCES `hjv`.`funktion` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_funktion_has_udir_udir1`
    FOREIGN KEY (`udir_id` )
```

```
    REFERENCES `hjv`.`udir` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`q`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`q` (
  `id` SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `q` INT UNSIGNED NOT NULL ,
  PRIMARY KEY (`id`) ,
  UNIQUE INDEX `q_UNIQUE` (`q` ASC) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`udir_giver_q`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`udir_giver_q` (
  `udir_id` SMALLINT UNSIGNED NOT NULL ,
  `q_id` INT UNSIGNED NOT NULL ,
  PRIMARY KEY (`udir_id`, `q_id`) ,
  INDEX `fk_udir_has_q_q1` (`q_id` ASC) ,
  INDEX `fk_udir_has_q_udir1` (`udir_id` ASC) ,
  CONSTRAINT `fk_udir_has_q_udir1`
    FOREIGN KEY (`udir_id` )
    REFERENCES `hjv`.`udir` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_udir_has_q_q1`
    FOREIGN KEY (`q_id` )
    REFERENCES `hjv`.`q` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`udir_indeholder_udir`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`udir_indeholder_udir` (
  `udir_id` SMALLINT UNSIGNED NOT NULL ,
  `udir_id1` SMALLINT UNSIGNED NOT NULL ,
  PRIMARY KEY (`udir_id`, `udir_id1`) ,
  INDEX `fk_udir_has_udir_udir4` (`udir_id1` ASC) ,
  INDEX `fk_udir_has_udir_udir3` (`udir_id` ASC) ,
  CONSTRAINT `fk_udir_has_udir_udir3`
    FOREIGN KEY (`udir_id` )
    REFERENCES `hjv`.`udir` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_udir_has_udir_udir4`
    FOREIGN KEY (`udir_id1` )
    REFERENCES `hjv`.`udir` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`fagplan`
```

```
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`fagplan` (
  `id` SMALLINT UNSIGNED NOT NULL ,
  `maal` TEXT NULL ,
  `formaal` TEXT NULL ,
  `indhold` TEXT NULL ,
  `dato` DATE NOT NULL ,
  `nummer` SMALLINT UNSIGNED NOT NULL ,
  `andet` TEXT NULL ,
  PRIMARY KEY (`id`) ,
  UNIQUE INDEX `nummer_UNIQUE` (`nummer` ASC) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`dokumenter`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`dokumenter` (
  `id` SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `indhold` TEXT NOT NULL ,
  PRIMARY KEY (`id`) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`udir_har_fagplan`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`udir_har_fagplan` (
  `udir_id` SMALLINT UNSIGNED NOT NULL ,
  `fagplan_id` SMALLINT UNSIGNED NOT NULL ,
  PRIMARY KEY (`udir_id`, `fagplan_id`) ,
  INDEX `fk_udir_has_fagplan_fagplan1` (`fagplan_id` ASC) ,
  INDEX `fk_udir_has_fagplan_udir1` (`udir_id` ASC) ,
  CONSTRAINT `fk_udir_has_fagplan_udir1`
    FOREIGN KEY (`udir_id` )
    REFERENCES `hjv`.`udir` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_udir_has_fagplan_fagplan1`
    FOREIGN KEY (`fagplan_id` )
    REFERENCES `hjv`.`fagplan` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`udir_har_dokumenter`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`udir_har_dokumenter` (
  `udir_id` SMALLINT UNSIGNED NOT NULL ,
  `dokumenter_id` SMALLINT UNSIGNED NOT NULL ,
  PRIMARY KEY (`udir_id`, `dokumenter_id`) ,
  INDEX `fk_udir_has_dokumenter_dokumenter1` (`dokumenter_id` ASC) ,
  INDEX `fk_udir_has_dokumenter_udir1` (`udir_id` ASC) ,
  CONSTRAINT `fk_udir_has_dokumenter_udir1`
    FOREIGN KEY (`udir_id` )
    REFERENCES `hjv`.`udir` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_udir_has_dokumenter_dokumenter1`
    FOREIGN KEY (`dokumenter_id` )
    REFERENCES `hjv`.`dokumenter` (`id` )
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`rolle`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`rolle` (
  `id` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `navn` VARCHAR(45) NOT NULL ,
  PRIMARY KEY (`id`) ,
  UNIQUE INDEX `navn_UNIQUE` (`navn` ASC) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`bruger_har_rolle`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`bruger_har_rolle` (
  `bruger_id` INT UNSIGNED NOT NULL ,
  `rolle_id` TINYINT UNSIGNED NOT NULL ,
  PRIMARY KEY (`bruger_id`, `rolle_id`) ,
  INDEX `fk_bruger_has_rolle_rolle1` (`rolle_id` ASC) ,
  INDEX `fk_bruger_has_rolle_bruger1` (`bruger_id` ASC) ,
  CONSTRAINT `fk_bruger_has_rolle_bruger1`
    FOREIGN KEY (`bruger_id` )
    REFERENCES `hjv`.`bruger` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_bruger_has_rolle_rolle1`
    FOREIGN KEY (`rolle_id` )
    REFERENCES `hjv`.`rolle` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`bruger_administrerer_kursus`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`bruger_administrerer_kursus` (
  `bruger_id` INT UNSIGNED NOT NULL ,
  `kursus_id` SMALLINT UNSIGNED NOT NULL ,
  PRIMARY KEY (`bruger_id`, `kursus_id`) ,
  INDEX `fk_bruger_has_kursus_kursus1` (`kursus_id` ASC) ,
  INDEX `fk_bruger_has_kursus_bruger1` (`bruger_id` ASC) ,
  CONSTRAINT `fk_bruger_has_kursus_bruger1`
    FOREIGN KEY (`bruger_id` )
    REFERENCES `hjv`.`bruger` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_bruger_has_kursus_kursus1`
    FOREIGN KEY (`kursus_id` )
    REFERENCES `hjv`.`kursus` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`bruger_leder_afdeling`
-- -----------------------------------------------------
```

```
CREATE  TABLE IF NOT EXISTS `hjv`.`bruger_leder_afdeling` (
  `bruger_id` INT UNSIGNED NOT NULL ,
  `afdeling_id` SMALLINT UNSIGNED NOT NULL ,
  PRIMARY KEY (`bruger_id`, `afdeling_id`) ,
  INDEX `fk_bruger_has_afdeling_afdeling1` (`afdeling_id` ASC) ,
  INDEX `fk_bruger_has_afdeling_bruger1` (`bruger_id` ASC) ,
  CONSTRAINT `fk_bruger_has_afdeling_bruger1`
    FOREIGN KEY (`bruger_id` )
    REFERENCES `hjv`.`bruger` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_bruger_has_afdeling_afdeling1`
    FOREIGN KEY (`afdeling_id` )
    REFERENCES `hjv`.`afdeling` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`bruger_har_funktion`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`bruger_har_funktion` (
  `bruger_id` INT UNSIGNED NOT NULL ,
  `funktion_id` SMALLINT UNSIGNED NOT NULL ,
  `prioritet` DECIMAL(2) UNSIGNED NOT NULL ,
  PRIMARY KEY (`bruger_id`, `funktion_id`) ,
  INDEX `fk_bruger_has_funktion_funktion1` (`funktion_id` ASC) ,
  INDEX `fk_bruger_has_funktion_bruger1` (`bruger_id` ASC) ,
  CONSTRAINT `fk_bruger_has_funktion_bruger1`
    FOREIGN KEY (`bruger_id` )
    REFERENCES `hjv`.`bruger` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_bruger_has_funktion_funktion1`
    FOREIGN KEY (`funktion_id` )
    REFERENCES `hjv`.`funktion` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `hjv`.`udir_kraever_udir`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`udir_kraever_udir` (
  `udir_id` SMALLINT UNSIGNED NOT NULL ,
  `udir_id1` SMALLINT UNSIGNED NOT NULL ,
  PRIMARY KEY (`udir_id`, `udir_id1`) ,
  INDEX `fk_udir_has_udir_udir2` (`udir_id1` ASC) ,
  INDEX `fk_udir_has_udir_udir1` (`udir_id` ASC) ,
  CONSTRAINT `fk_udir_has_udir_udir1`
    FOREIGN KEY (`udir_id` )
    REFERENCES `hjv`.`udir` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_udir_has_udir_udir2`
    FOREIGN KEY (`udir_id1` )
    REFERENCES `hjv`.`udir` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `hjv`.`afdeling_underafdeling_af_afdeling`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `hjv`.`afdeling_underafdeling_af_afdeling` (
  `afdeling_id` SMALLINT UNSIGNED NOT NULL ,
  `afdeling_id1` SMALLINT UNSIGNED NOT NULL ,
  PRIMARY KEY (`afdeling_id`, `afdeling_id1`) ,
  INDEX `fk_afdeling_has_afdeling_afdeling2` (`afdeling_id1` ASC) ,
  INDEX `fk_afdeling_has_afdeling_afdeling1` (`afdeling_id` ASC) ,
  CONSTRAINT `fk_afdeling_has_afdeling_afdeling1`
    FOREIGN KEY (`afdeling_id` )
    REFERENCES `hjv`.`afdeling` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_afdeling_has_afdeling_afdeling2`
    FOREIGN KEY (`afdeling_id1` )
    REFERENCES `hjv`.`afdeling` (`id` )
    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;
```