Master i IT

Masterprojekt, Softwarekonstruktion

AALBORG UNIVERSITET

Omstilling til SAP HANA SQLScript – en "Discount" undersøgelse

Mikkel Vad Damgaard
Oktober 2020

Vejleder:
Professor (MSO) Bent Thomsen

Indhold

Abstract

The main objective in this Master Thesis is to determine whether the "Discount Method for Programming Language Evaluation"[1] in a specific case from the Danish Defence can be used for evaluating the need for education when computer programmers shift away from one programming language and are introduced to a new programming language.

The "Discount Method" was invented in order to facilitate a simple method for testing a new computer language under its development as a cost effective way to evaluate strengths and weaknesses several times during the development process of the new language.

During their carreers, computer developers will often shift from one programming language to another. A shift can be caused by the need of implementing new technology or it can be caused by coincidences e.g. when two departments are merged. It seems inevitable that a shift will result in some extra costs – at least in the short term.

The focus of the Thesis is to analyse whether the "Discount Method for Programming Language Evaluation" can be used in this scope.

Computer Programmers working for the Danish Defence are used as a case study for the analysis. The Danish Defence has a SAP based Enterprise Resource and Planning system including a large Business Intelligence system.

New database technology has recently been introduced in the Business Intelligence system, hence the developers will be forced to move to a new programming language called SAP HANA SQLScript.

In the Thesis it is concluded that the "Discount Method" can be used to determine the need for education when changing from one programming language to another – in this case SAP HANA SQLScript. However it is also covered that the ease of use depends on the existence of the prerequisites for the method, where especially the "Cheat Sheet" and the "Task Sheet" are vital components. If these do not exist in advance, use of the method will be some more cumbersome.

[1] Developed at Aalborg University and described in [1].

1

Indledning

Dette projekt handler om at tilpasse Forsvaret til en større teknisk omlægning af it-systemet DeMars. De fleste større omlægninger af it-systemer medfører en eller anden form for uddannelse af personalet, som betjener systemerne - ikke mindst de personer, som vedligeholder og videreudviklinger systemerne. Projektet undersøger en metode til at vurdere uddannelsesbehovet ved programmørerne, når disse skal omstilles fra et programmeringssprog til et andet. Den aktuelle opgradering af DeMars Business Intelligence udgør den konkrete case for projektet.

Det danske Forsvar har gennem de seneste ca. 20 år anvendt en SAP[1] baseret løsning benævnt DeMars til styring af ressourcer, herunder blandt andet økonomi, personale og logistik. I den samme periode er der gennemført en samling af ensartede administrative funktioner fra decentrale funktioner til centrale styrelser. Gennem de senere år er der således samlet funktioner og oprettet styrelser som fx Forsvarsministeriets Regnskabsstyrelse, Forsvarsministeriets Personalestyrelse og Forsvarsministeriets Materiel- og Indkøbsstyrelse.

Omstruktureringerne er sket i tæt samspil mellem de organisatoriske valg og de it-systemmæssige afhængigheder, hvor det har været en forudsætning, at DeMars skal anvendes som det primære værktøj, og at processer så vidt muligt skal være ens på tværs i koncernen.

Som en konsekvens har Forsvarsministeriet valgt at samle flest mulige funktionaliteter i SAP systemet DeMars, og gøre systemet til et fælles "Enterprise Ressource and Planning" (ERP) værktøj. Igennem årene har det været tilgangen, at Forsvaret med sit valg af et SAP system skulle følge med udviklingen inden for SAP - uden nødvendigvis at skulle være "first mover". DeMars er på den måde flere gange blevet opgraderet, når situationen har været moden, med det er også flere gange sket i forbindelse med omstruktureringer fx i forbindelse med politisk indgåede forsvarsforlig.

Når en større omstilling sker som følge af en organisatorisk udvikling, medfører det typisk ændringer til datastrukturer som fx organisatoriske hierarkier eller æn-

[1] SAP®: Systems, Applications and Products in Data Processing.

dringer til eksisterende funktionaliteter afledt af de nye processer. Ændringerne kan oftest gennemføres i de eksisterende systemer, værktøjer og programmeringssprog.

De kommende større ændringer i DeMars forventes i modsætning til ovenstående at ske som følge af nye teknologiske muligheder. Denne type af ændringer medfører ofte også ændringer til programmørernes brugergrænseflader i form af nye værktøjer eller programmeringssprog. I projektet undersøges det, om omfanget af den nødvendige omstilling og uddannelse af programmørerne kan fastlægges ved hjælp af "Discount Method of Programming Language Evaluation" udviklet ved Aalborg Universitet. Undersøgelsen sker som en komparativ analyse mellem "The Cognitive Dimensions of Notation Framework" og "Discount Method of Programming Language Evaluation".

2

Baggrund

2.1 Omkostninger ved skift af programmeringssprog

Når en virksomhed eller organisation vælger at udskifte ét programmeringssprog med et andet, kan det være forbundet med store omkostninger. Et nyt programmeringssprog betyder typisk, at tidligere retningslinjer for programmering og udvikling skal revurderes eller udarbejdes på ny. Det kan fx betyde ny standard for kodesyntaks, test og dokumentation af kode.

Det vil som oftest også betyde, at programmørerne, testere med flere skal uddannes i det nye sprog.

Mange virksomheder og organisationer, har allerede en omfattende mængde af kode skrevet i det sprog, der står for udskiftning til fordel for et nyt sprog. Heri ligger der en udfordring i at vurdere, om eksisterende kode skal konverteres til det nye sprog eller forblive, som den er – med den konsekvens, at der nu reelt er to sprog i brug. Hvis både gammelt og nyt sprog fastholdes, betyder det, at de "gamle" programmører skal uddannes i det nye sprog, men også at programmører, der ansættes fremadrettet, vil skulle uddannes i det gamle sprog.

Uanset hvilken tilgang der vælges ved introduktion af nyt computersprog, synes det oplagt, at det vil medføre en række omkostninger, der enten vil være kortvarige eller belaste økonomien i en længere periode. I lyset heraf er det nærliggende at antage, at der er forsket meget i fordele og ulemper ved at introducere et nyt sprog. Det ses imidlertid ikke at være tilfældet "However, research suggests that it isn't the case in software engineering, especially with respect to programming design"[2].

2.2 Hvorfor skifte programmeringssprog?

Hvis det er forbundet med omkostninger at skifte fra et computersprog til et andet, hvorfor så overhovedet gøre det?

I nogle tilfælde kan det måske bero på tilfældigheder. Det sker fx når to virksomheder eller to afdelinger i én virksomhed sammenlægges, og den ene parts praksis videreføres for alle.

I andre sammenhænge kan der være truffet et strategisk valg om hele tiden at være "up-to-date". Det kan fx være et led i en HR-strategi om løbende at kunne tiltrække og fastholde kvalificerede programmører.

Endeligt kan der være tilfælde, hvor der tages helt ny teknologi i anvendelse. De fleste i it-verdenen vil nok kunne nikke genkendende til, at internettets udbredelse har medført ny teknologi og hermed nye computersprog - tænk blot på html og Java.

Dette projekt handler netop om det at være nødt til at skifte computersprog som følge af et ønske om at udnytte en ny teknologi - nemlig til brug af sproget SAP HANA SQLScript, som netop er den udfordring, Forsvaret står over for. Projektets fokus er, hvordan uddannelsesbehovet hos programmørerne kan fastlægges.

3

Formalia

Masterprojektet er udarbejdet inden for rammerne af Studieordning for Masteruddannelsen i IT, Linjen i Softwarekonstruktion[3]. Forud for masterprojektet har forfatteren gennemført:

- Fagpakke i Netværksikkerhed
- Fagpakke i IT-sikkerhed i organisationer
- Fagpakke i Data Science og Big Data

Masterprojektet følger formalia i Studieordningen[3], herunder er projektet udarbejdet på dansk, idet resuméet dog er skrevet på engelsk jf. Studieordningens §20.

Numre indsat i firkantede parenteser [ ] henviser til Kilder, der findes bagest i projektet.

Den resterende del af projektet består af følgende kapitler:

- **Kapitel 4** indeholder en kort beskrivelse af DeMars, der er Forsvarets ERP system. DeMars er et meget komplekst system. Formålet med kapitlet er at give læseren mulighed for at sætte den øvrige del af projektet i kontekst.
- **Kapitel 5** indholder problemformuleringen og en kort diskussion heraf.
- **Kapitel 6** indholder resultaterne af en analyse af forskellige muligheder for at opstille et HANA-miljø til brug for gennemførelse af projektet og senere uddannelse af programmører.
- **Kapitel 7** indeholder en kort introduktion til sproget SAP HANA SQLScript, herefter analyseres sproget ved hjælp af en operationalisering af rammeværket Cognitive Dimensions of Notation Framwork.

- **Kapitel 8**. I kapitlet gennemgås kort "Discount Method for Programming Language Evaluation".
- **Kapitel 9**. I kapitlet beskrives det, hvordan "Discount Method for Programming Language Evaluation" har været anvendt til at undersøge uddannelsesbehovet for en række programmører i forbindelse med deres kommende omstilling til sproget SAP HANA SQLScript. Herefter opsummeres resultaterne af den faktiske analyse, der findes i bilag 4.
- **Kapitel 10** indeholder en sammenstilling af resultaterne fra brugen af "Cognitive Dimensions" i kaptitel 7 med resultaterne fra brugen af "Discount Method for Programming Language Evaluation" i kapitel 9. Kapitlet afsluttes med en opsamling på projektets delkonklusioner og den samlede konklusion.
- **Kapitel 11** indeholder en kort drøftelse af nogle af de mulige usikkerheder i forhold til projektets konklusion.
- **Kapitel 12**. Projektet afsluttes i kapitel 11, idet der ses på, hvordan projektets resultater vil kunne anvendes i Forsvaret fremadrettet, ligesom der reflekteres over nogle praktiske erfaringer fra gennemførelsen af projektets forsøg.

Masterprojektet er skrevet i LaTeX.

I forbindelse med udarbejdelse af projektet er der gennemført en række forsøg. Forsøgene er grundlaget for projektets resultater, og optagelse af forsøgene i form af skærmaktivitet og tale er vedlagt. Disse skal behandles jf. gældende regler for GDPR, herunder skal de destrueres, når de ikke længere opfylder deres formål i relation til masterprojektet.

4

DeMars

Dette kapitel giver en kort introduktion til DeMars. Kapitlet rummer det netop nødvendige for at læseren kan få indblik i den kontekst, projektet indgår i. DeMars har været under konstant udvikling i Forsvaret gennem de seneste 20 år. Kapitlets fokus er baggrunden for den udvikling, der pågår netop nu og den udvikling, der kan forudses i de kommende år. En udvikling, der i høj grad har betydning for brug af programmeringssprog og hermed den softwareudvikling, der løbende finder sted.

4.1 Formålet med DeMars

DeMars er Forsvarets Enterprise Ressource and Planning (ERP) system, der anvendes til stort set alt administration af Forsvarets ressourcer. Det er i DeMars, at Forsvarets årlige bevilling på ca. 23,6 mia. kr.[4] styres.

DeMars bruges fx til administration af ca. 50.000 personer, herunder lønudbetaling til de mere end 20.000 fastansatte. Der styres over 2.000 køretøjer, ca. 60 større skibe og lidt over 100 luftfartøjer i DeMars. Forsvaret indkøber materiel for ca. 7 mia. kr. hvert år. Det dækker alt fra snørebånd til nye kampfly. Alt indkøbes gennem DeMars, og lagrene, der er spredt i ind- og udland, omfatter ca. 400.000 varenumre, som også indgår i DeMars.

DeMars er bygget på en SAP platform, der regnes for den mest udbredte ERP platform blandt større vestlige organisationer og virksomheder. Noget forsimplet består DeMars af en omfattende database (der af sikkerhedsmæssige hensyn er replikeret over flere geografiske lokationer) og et stort antal klienter (ca. 15.000 brugere), der kan tilgå, sammenstille, vedligeholde og skabe nye data.

Databasen har traditionelt haft sin egen databaseserver. Ved siden af denne er der en række applikationsservere, hvor programmerne ligger. Databasen har hidtil kunnet være én blandt mange SQL (Structured Query Language) databaser som fx Microsoft® SQL Server, IBM® OpenDB eller Oracle®[1]. Programmerne på applikationsserverne kommunikerer med databasen gennem SQL. Brugerne kommunikerer

[1] Oracle har været den foretrukne database til større SAP installationer.

med programmerne gennem en GUI (Graphical User Interface) på deres standard PC arbejdsplads. På trods af navnet, kan GUIen reelt kun præsentere og udveksle data i form af tal og tekst – ikke grafik.

DeMars er et transaktionssystem, hvor input fra brugerne gemmes som selvstændige transaktioner, og nogle få saldi ajourføres. Databasen er opbygget som en relationsdatabase med et meget stort antal tabeller.

Det betyder fx, at det er muligt at se samtlige varebevægelser ind og ud af et lager samt aktuel lagerbeholdning. Der findes imidlertid ikke informationer om, hvordan lagerbeholdningen så ud for nogle år siden. Denne oplysning må i stedet beregnes ved at tage aktuel lagerbeholdning og regne baglæns for alle varebevægelser fra de forgange år.

Det kan være omstædigt at udlede data, der dækker over mange tabeller, og som har en tidsmæssig udstrækning. For at imødegå dette, er der indført et Business Intelligence (BI) system som en integreret del af DeMars.

4.2 Business Intelligence i DeMars

BI systemet er et beslutningsstøttesystem. I BI systemet kan aktuelle og historiske data sammenstilles på tværs af tabeller, analyseres og præsenteres til brug for den løbende sagsbehandling, der sker i styrelserne under Forsvarsministeriet, og som danner grundlag for ledelsesmæssige beslutninger.

For at gøre det muligt ekstraheres der hver nat data fra databasen. Data kombineres på tværs af deres oprindelige tabeller, beriges med yderligere beregninger og forretningslogik og præsenteres i særlige BI værktøjer, hvor muligheder for at bruge figurer, grafer m.m. er bedre end i GUIen.

En væsentligt ulempe ved BI er, at den natlige ekstrahering af data foregår gennem en række prædefinerede loads. Selv beskedne ændringer til data i en rapport kan derfor tage flere arbejdsdage at implementere, og helt nye løsninger kan være både uger og måneder under vejs.

4.3 Udvikling af DeMars

Der er løbende sket udvikling af DeMars igennem årene. Udvikling dækker over alt fra banale fejlrettelser og enkle brugerønsker til store it-udviklingsprojekter til trecifrede millionbeløb. Hertil kommer der i sagens natur løbende opdateringer af operativsystemer og sikkerhedsopdateringer. I gennemsnit har der gennem årene været ansat ca. 80 personer til at drifte og videreudvikle DeMars. Blandt disse har der været ca. 30 programmører.

Programmørerne har langt overvejende arbejdet med software til applikationsserverne. Det mest anvendte programmeringssprog har været ABAP (Advanced Business Application Programming[2]).

Når der er behov for, at programmerne på applikationsserveren kommunikerer med databasen, sker det gennem SQL indlejret i ABAP. Dette billede har stort set ikke ændret sig igennem de ca. 20 år med DeMars.

[2]Oprindeligt: "Allgemeiner Berichts-Aufbereitungs-Prozessor".

4.3.1 In-memory teknologi

Igennem de senere år er prisen på RAM hukommelse (Random Acces Memory) faldet drastisk. Tidligere blev data opbevaret på disk. Data blev kun overført til RAM, når der skulle udføres operationer på data. Umiddelbart efter operationens gennemførelse blev dennes resultat skrevet tilbage til disk.

Med den billige og hermed større adgang til RAM, har billedet ændret sig. Spark er måske mest kendte eksempel inden for databaser til store mængder af data. Spark har mange steder afløst Hadoop teknologien[5].

I Hadoop gemmes alle resultater inklusiv mellemresultater på disk, det betyder, at hvis processen A og B skal gennemføres i rækkefølge på nogle data, sker det ved: at data først hentes fra disk til proces A, efter processen er gennemført, gemmes resultatet igen på disk, umiddelbart herefter hentes resultaterne fra proces A på ny fra disk, hvorefter proces B gennemføres, og det endelige resultat gemmes på disk.

I Spark gemmes resultatet fra proces A i hukommelsen, og det kan derfor indgå direkte i proces B uden den langsommelige mellemfaldende diskoperation.

En lignende tendens ses inden for SAP, hvor SAP har udviklet databasen SAP HANA® (high-performance analytic appliance). Set fra et konceptuelt perspektiv er brugen af RAM i HANA endnu mere udtalt end i eksemplet fra Spark.

I HANA er hele databasen som udgangspunkt opbevaret i RAM.[3] Ifølge SAP er den "rå" hastighedsforøgelse ved brug af HANA-teknologien i forhold til traditionelle diskbaserede databaser omkring faktor 1000.

HANA databaser er bagud kompatible. De kan altså bruges som enhver anden SQL database, hvor svartiden blot er meget hurtig.

4.3.2 Code push down

HANA teknologien er imidlertid meget mere end en database, hvor data er placeret in-memory[6]. Hvor en traditionel SQL database tillader forespørgsler på data i form SQL kald fra en applikation eller i form af SQL Stored Procedures, der er placeret på databasen, stiller HANA databasen øget funktionalitet til rådighed.

HANA muliggør, at der kan placeres kode på databaseserveren, som eksekveres på serveren ligesom SQL Stored Procedures, men som giver væsentligt mere fleksible muligheder, end det Stored Procedures tilbyder. SAP kalder flytning af kode fra applikationsserveren til databasen for "Code Push-down".

Med Code Push-down er det muligt at udføre komplekse beregninger og analyser direkte på databaseserveren som fx:

- Predictive analytics
- Spatial data processing
- Text analytics
- Text search
- Streaming analytics
- Graph data processing

Den kode, der placeres og udføres på HANA serveren, skrives i SAP HANA SQLScript[4] – et sprog, SAP har udviklet til netop dette formål.

[3] Af hensyn til redundans m.m. sker der en løbende replikering af data til disk, så systemet vil kunne gendannes (til et kendt tidspunkt) i tilfælde af systemnedbrud.

[4] Anføres ofte blot som "SQLScript", hvis det ikke giver anledning til misforståelser.

4.3.3 Brug af HANA teknologi til Business Intelligence

I skrivende stund er Forsvaret ved at afslutte udskiftningen af databaserne fra Oracle til HANA. Udskiftningen sker som "lift and shift", hvor det udnyttes, at HANA teknologien er bagud kompatibel med SQL databaser. Adgangen til data er blevet væsentligt hurtigere, men programmørerne programmerer fortsat mod applikationsserverne, hvorfra det er nøjagtig de samme SQL forespørgsler, der udføres mod databasen som tidligere. Potentialerne i Code Push-down udnyttes altså pt. ikke.

I forhold til Business Intelligence loades der derfor fortsat data hver nat, og data sammenstilles i de kuber, der bruges i de prædefinerede analyser og ledelsesrapporter. En af de væsentligste bevæggrunde for at indføre HANA til Business Intelligence var at kunne komme ud over begrænsninger ved natlige loads og i stedet kunne sammenstille data i realtid. Potentialerne udnyttes således heller ikke på BI området. BI programmørerne skal derfor nu i gang med at flytte eksisterende programmer fra applikationsserverne til HANA databaserne, ligesom ny udvikling skal gennemføres på HANA databaserne. Det er derfor særdeles nødvendigt, at programmørerne omstilles fra ABAP til SAP HANA SQLScript.

5

Problemformulering

Forsvarets aktuelle situation er beskrevet i forrige kapitel. I forlængelse af den formuleres følgende problemstilling.

Kan "Discount Method for Programming Language Evaluation" bruges til at analysere og fastlægge uddannelsesbehovet ved overgangen fra et programmeringssprog til et andet?

Case: programmørerne ved Forsvaret DeMars Business Intelligence skal omstille sig fra programmering i SAP Advanced Business Programming Language (ABAP) og standard SQL til SAP HANA SQLScript inden for det kommende år. I den forbindelse:

1. undersøges det og beskrives kort, hvordan et analyse-/uddannelsesmiljø for SAP HANA SQLScript kan tilvejebringes med henblik på både at kunne indgå i analyserne i de to øvrige punkter nedenfor og senere vil kunne anvendes til uddannelse af programmører.

2. gennemføres der en analyse af SAP HANA SQLScript efter en anerkendt analysemodel, således at resultatet kan anvendes til at validere resultaterne af tredje punkt nedenfor.

3. analyseres det ved forsøg, om "Discount Method for Programming Language Evaluation" kan anvendes til at vurdere behovet for uddannelse, der er nødvendigt, for at kunne omstille programmørerne fra ABAP og standard SQL til SAP HANA SQLScript.

5.1 Opgaveanalyse

Det første underpunkt i problemformuleringen foreskriver, at det kort skal beskrives, hvordan et analyse-/uddannelsesmiljø for SAP HANA SQLScript kan tilvejebringes.

Ordlyden lægger op til en redegørelse, men da projektet udarbejdes inden for linjen Softwarekontruktion, gennemføres der faktisk opsætning og en beskrivelse, som Forsvaret efterfølgende vil kunne anvende. En del af den medgåede tid til projektet vil således have praktisk karakter. Det bemærkes dog, at det er anført, at beskrivelsen skal være kort. Endelig ses et succesfuldt resultat af første underpunk at være en forudsætning for analyserne i de to øvrige underpunkter.

Det andet underpunkt i problemformuleringen anfører, at der skal der vælges en anerkendt analysemodel til evaluering af SAP HANA SQLScript, således at dennes resultater kan sammenholdes med resultaterne fra forsøgene gennemført under Discount Metoden med henblik på validering. Det er ikke yderligere defineret, hvad analysemodellen skal opfylde – udover at være anerkendt.

I forhold til det tredje underpunkt bemærkes det, at det af problemformuleringens case fremgår, at projektet drejer sig om at vurdere omstillingen af allerede eksisterende BI programmører ved Forsvaret fra ABAP til SAP HANA SQLScript.

Det er altså disse programmørers eksisterende færdigheder og kompetencer, der vurderes i forhold til de fremtidige behov inden for SAP HANA SQLScript. I praksis betyder det, at de eksisterende kompetencer og ikke mindst forskelle i kompetenceniveau må accepteres, som de er.

Herudover bemærkes det, at det eksplicit er udtrykt, at projektets gennemførelse skal ske ved forsøg.

6

Miljø til analyse og uddannelse

6.1 Konfigurering af et SAP HANA database miljø til analyse og uddannelse

Ifølge problemformuleringens første underpunkt skal det undersøges og kort beskrives, hvordan et analyse-/uddannelsesmiljø for SAP HANA SQLScript kan tilvejebringes til brug for yderligere analyser i projektet og senere til uddannelse i Forsvaret.

Den umiddelbare løsning kunne være at bestille miljøet hos Forsvarets driftsleverandør for DeMars. Det vil imidlertid betyde, at brugerne er afhængige af adgang til Forsvarets lukkede netværk, hvorfra der ikke kan opnås adgang fra internettet. Uddannelse, selvstudie m.m. vil således kræve fysisk adgang til Forsvarets netværk, hvilket vil begrænse brugernes mulighed for at anvende miljøet fleksibelt. I analyse og uddannelsesmæssig sammenhæng vil der ikke være behov for adgang til klassificerede data, som retfærdiggør denne adskillelse, som ellers er obligatorisk i Forsvaret. Hertil kommer, at indkøb og drift af et miljø andrager ca. kr. 25.000 pr. måned altså en årlig udgift på kr. 300.000.

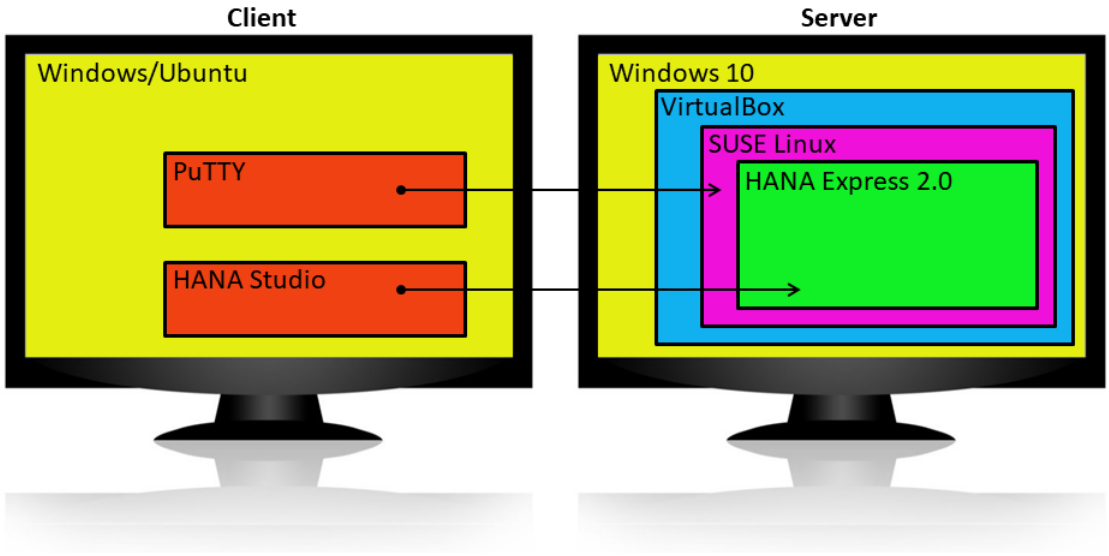
Det er således ønskeligt at undersøge mulighederne for at opsætte og anvende en mere fleksibel og billigere løsning.

6.1.1 Konfiguration af miljø

SAP tilbyder SAP HANA Express 2.0 databasen til brug for kundernes test, analyse og uddannelse på HANA teknologien. Databasen har den fulde SAP HANA funktionalitet. Den eneste praktiske begrænsning ligger i mængden af data, der kan opbevares i databasen, der er begrænset til 32 GB data. SAP HANA Express kan fås til en bred vifte af hardware og styresystemer, og den kan således også afvikles på en PC platform.

Da det netop er kendetegnet for HANA teknologien, at databasen er placeret i RAM, kræver det en kraftig PC for at kunne rumme databasen.[1] Forsvaret anvender

[1] Minimum 24 GB RAM. SAP anbefaler 32 GB.



Figur 6.1: Udviklingsopsætning

hovedsagligt bærbare PCer, og særligt her kan det være svært at honorere kravet om mængden af RAM. Det har dog vist sig muligt.

Flere kombinationer af hardware, styresystemer og installationer har været afprøvet i forbindelse med projektet, herunder stationære PCer, bærbare PCer, windows 10, Ubuntu Linux, SUSE Linux, binære installationer af SAP HANA Express og brug af Virtual Box images.

Den anbefalede konfiguration fremgår af figur 6.1. Figuren viser, at der anvendes to separate computere i opsætningen. I praksis har det vist sig muligt at lade én og samme PC udgøre både server og client. SAP fraråder dog dette, og ved at adskille server og clients på forskelligt hardware er det også muligt at koble flere klienter på den samme server - med andre ord kan flere elever uddannes samtidigt under anvendelse af én server. Det bemærkes, at en almindelig hæderlig bærbar PC fint kan anvendes som client.

Det viste sig mest praktisk at anvende en kraftig bærbar PC som server. Computeren var bestykket med en Pentium I7 processor, 32 GB RAM og en 512 MB Solid State harddisk. Der blev anvendt Windows 10 som det primære styresystem. Herpå blev der installeret Oracle VirtualBox. På Virtual Box installeres et image fra SAP, som indeholder en prekonfigureret SAP HANA Express 2.0 database på et SUSE Linux Enterprise styresystem.

SAP leverer også installationsvejledning til binær installation af HANA Express 2.0 uden brug af virtualisering. Det var en meget tung proces i forhold til både installation og efterfølgende opsætning, og i praksis var der ikke synlige forskelle i fx funktionalitet eller performance i forhold til et virtual image. Arbejdet med den "rigtige" installation tog flere dage, mens en virtuel installation kunne gennemføres på nogle timer.

Det kan ikke udelukkes, at der i forbindelse med en tung brug af databasen vil være forskel i performance for en "rigtig" installation og en virtuel installation.

På clientsiden blev der anvendt både en PC med MS Windows og en PC med

Ubuntu Linux. På begge clienter blev der installeret PuTTY og HANA Studio[2]. De to clienter var trods forskellige styresystemer helt ens i funktionalitet.

HANA Studio blev anvendt til al kommunikation mellem clienten og databasen HANA Express 2.0. Start, stop og overvågning af databasen sker direkte fra styresystmet, der i dette tilfælde er SUSE Linux. Det kan gøres fysisk direkte på serveren. I praksis viste det sig dog at være nemmere at anvende en SSH forbindelse (PuTTY) fra clienterne til serveren.

6.1.2 Delkonklusion

I dette afsnit er det kort beskrevet, hvordan der kan etableres et SAP HANA miljø til brug for yderligere analyse af og undervisning i HANA teknologi. Den viste opsætning giver fuld funktionalitet og en performance, som har vist sig ikke at begrænse mulighederne for yderligere analyser og undervisning. Da etableringen ikke medfører drifts- eller licensomkostninger, er den eneste udgift anskaffelse af en kraftig dedikeret server PC. Udgiften til anskaffelsen af denne udgør maksimalt 15.000 kr. altså mindre end en måneds drift af et tilsvarende miljø fra DeMars driftsleverandøren.

Det er således undersøgt og kort beskrevet, hvordan et analyse-/uddannelsesmiljø til SAP HANA SQLScript kan tilvejebringes med henblik på både at kunne gennemføre de øvrige analyser i masterprojektet og senere vil kunne anvendes til uddannelse af programmører. Med andre ord er problemformuleringens første underpunkt (Miljø) besvaret.

[2]HANA Studio er Eclipse med nogle SAP specifikke plug-ins.

7

SAP HANA SQLScript

I dette kapitel gives der en kort introduktion til SAP HANA SQLScript. Det er ikke hensigten at gennemføre en gennemgribende præsentation af SAP HANA SQLScript som sprog, dets syntaks eller sprogets fordele og ulemper i forhold til andre sprog. Det er heller ikke hensigten, at kapitlet skal kunne fungere som en dækkende reference til brug af sproget. En kort oversigt over det væsentligste indhold i SQLScript findes derimod i Bilag C - Cheat Sheet, mens en mere grundig beskrivelse fremgår af [7].

Kapitlets formål er derimod:

- at beskrive, at SAP HANA SQLScript rent faktisk byder på nogle nye muligheder, som retfærdiggør den investering, det er at introducere et nyt computersprog i Forsvaret.
- kort at introducere læseren til en overordnet gruppering af programmeringssprog og herefter vise, at SAP HANA SQLScript indeholder elementer fra flere af grupperne på samme tid.
- at gennemføre en analyse af SAP HANA SQLScript ved hjælp Cognitive Dimensions fra et uddannelsesmæssigt perspektiv.

7.1 Formålet med SAP HANA SQLScript[1]

Som tidligere beskrevet kan der være flere formål med at indføre et nyt computersprog, og måske sker det lidt tilfældigt.

Det er derfor relevant at se på, hvad leverandøren SAP selv angiver som formålet med SAP HANA SQLScript, og herunder at undersøge om sproget ser ud til at indfri formålet.

[1] Dette afsnit er baseret på [7].

Ifølge SAP er formålet med SAP HANA SQLScript at indlejre dataintensive applikationer i HANA databasen. Hidtil har kode overvejende været afviklet på en applikationsserver, og kun relativ begrænset funktionalitet har været overladt til databasen gennem brug af SQL. Det har betydet, at store mængder data har skulle flyttes frem og tilbage mellem databasen og applikationsserverne, hvilket har været en omkostningsfuld proces målt i overførselstid og processorforbrug.

Når logikken på applikationsserveren skrives i fx ABAP, er der en tendens til at programmøren skriver kode, som gennemløber og behandler mængden af dataset sekventielt, hvilket besværliggør optimering og parallelisering i et multiprocessormiljø.

SAP HANA databasen er konstrueret med henblik på at udnytte de muligheder, der er i moderne hardware, herunder at gemme det komplette omfang af data i RAM og udnytte multi-core processorer. I praksis har SAP suppleret OpenSQL med en række udvidelser – samlet benævnt SAP HANA SQLScript[2].

SQLScript er blandt andet kendetegnet ved:

- Programmøren kan definere og anvende lokale variable, der fx kan indeholde strukturer eller tabeller, herunder resultater af mellemregninger.
- Koden kan inddeles i logiske containers i form af procedurer (sekvens af datatransformationer), Scalar User Defined Functions (der returnerer en scalar) og Table User Defined Functions (der returnerer en tabel).
- Det er muligt at anvende traditionel SQL i form af Data Definition Language udtryk (fx `CREATE Schema`), Data Manipulating Language udtryk (fx `INSERT`) og forespørgsler (fx `SELECT`).
- Der er muligt at styre kodens eksekvering gennem brug af betingelser (fx. `IF-ELSE`) og loops (fx `FOR`).
- Procedurer kan returnere flere resultater i modsætning til SQL, der kun kan returnere ét.

SAP anfører[3], at det med SAP HANA SQLScript er muligt at "implement applications by using both imperative orchestration logic and (functional) declarative logic". Herudover beskrives[4] SAP HANA SQLScript som et "Procedural" computersprog.

7.2 Deklarative og imperative sprog[5]

For nærmere at kunne undersøge om SAP lever op til sit eget formål med SAP HANA SQLScript, må det først klarlægges, hvad der forstås ved henholdsvis et deklarativt og et imperativt sprog.

I et deklarativt sprog skriver programmøren, hvad resultatet af koden skal være, ikke hvordan resultatet skal fremkomme. SQL er et typisk deklarativt sprog[6],

[2] I SAP kontekst benævnes sproget ofte blot SQLScript.
[3] [7] p.11.
[4] [7] p.7.
[5] Dette afsnit er baseret på [8].
[6] Frit oversat fra [8].

hvor programmøren i SQL syntaksen fx beskriver de data, der skal komme ud af en forespørgsel. Herefter overlader programmøren det til databasen at oversætte forespørgslen til intern eksekverbar kode.

Programmøren kan fx finde efternavnet på alle ansatte, der har en månedsløn på under 40.000 kr. ved at skrive denne SQL kode.

```
1 SELECT Efternavn FROM Ansatte WHERE (Loen<40000);
```

Listing 7.1: Traditionel SQL - et deklarativt sprog.

Som det ses, kan programmøren i eksemplet nøjes med at skrive én linje kode og skal således ikke bekymre sig om, hvordan koden efterfølgende udføres i databasen.

Som en modsætning til deklarative computersprog findes de imperative sprog. Det er de sprog, hvor programmøren omhyggeligt skal styre programmets forløb. Mange har nok stiftet bekendtskab med et imperativt sprog som fx Basic eller C++, hvor det er nødvendigt linje for linje at beskrive, hvad programmet skal gøre for at indfri et bestemt mål. Hvis man har gjort sig erfaringer med et af disse sprog, er det sandsynligt, at man også har debugget sig igennem dele af koden, og set hvilken konsekvens hver enkelt kodelinje har.

Til at styre programmets forløb har programmøren fx kontrolstrukturer, hvor de mest kendte måske er IF-THEN-ELSE, der er gennemgående i flertallet af imperative sprog. Også mulighederne for at gentage en blok af kode, et bestemt antal gange eller indtil en betingelse er opfyldt, kan genfindes i mange imperative sprog - de kendes fx som `FOR-NEXT` løkker eller `REPEAT-UNTIL` løkker.

De imperative computersprog er i hovedreglen også procedurale. Det procedurale består i, at koden inddeles i logiske blokke af kode, der eventuelt kan kaldes med et antal parametre og i nogle tilfælde enten returnerer et resultat eller ændringer på parametrene. I flere sprog kendes disse som procedurer og funktioner.

I [8] er forskellen mellem et proceduralt og et deklarativt sprog angivet kort og beskrivende: “Procedural languages result in many lines of code. Declarative languages result in one statement of the desired result”.

7.3 The Cognitive Dimensions of Notations Framework

For at opnå et bedre kendskab til SAP HANA SQLScript gennemføres en indledende analyse af sproget. Analysen gennemføres inden for metoden “The Cognitive Dimensions of Notations Framework”. Det er ikke i sig selv en analysemetode men derimod et rammeværk, der kan anvendes til at beskrive et system beregnet til notation af informationer i bred forstand. Ved den brede tilgang opnås, at Cognitive Dimensions fx kan anvendes til at beskrive og vurdere notationsformen i strikkeopskrifter og musiknoder - men også i programmeringssprog til computere.

Kernen i Cognitive Dimensions er en række dimensioner, der er centreret om brugerenes oplevelse af notationen. Dimensionerne omfatter fx hvor meget tankevirksomhed notationen kræver af brugeren (Hard Mental Operations), hvor nemt det er at udføre rettelser i notationen (Viscosity), eller om ensartede formål også beskrives og behandles ensartet (Consistency).

Cognitive Dimensions blev beskrevet sidst i 1980'erne (Green 1989) og videreudviklet op gennem 1990'erne. Antallet af og navnene på dimensionerne har ændret sig i takt med udviklingen.

7.3.1 Cognitive Dimensions i praksis

I 2007 beskrives en praktisk anvendelse af Cognitive Dimensions[9], hvor de brede anvendelsesmuligheder fastholdes, men hvor der samtidigt udarbejdes et spørge- eller interviewskema, som hjælper til at belyse de nu 14 dimensioner i rammeværket. Artiklen udvider således rammeværket med en konkret metode. Artiklen omfatter også en pilotafprøvning af spørge- eller interviewskemaet. I evalueringen heraf anføres det blandt andet, at når metoden anvendes til at beskrive et computersprog, så kan det være svært for programmørerne at skelne mellem det faktiske sprog og den editor eller IDE, det anvendes i.[7]

Gennem årene er Cognitive Dimensions blevet behandlet i en lang række papers og ved internationale konferencer ud fra både teoretiske og praktiske perspektiver[8]. Rammeværket og de praktiske implementeringer heraf må således siges at være anerkendte og dermed leve op til kravet i problemformuleringens andet underpunkt om at tilvejebringe et valideringsgrundlag gennem en anerkendt analysemodel.

7.3.2 Cognitive Dimensions og computersprog

Den centrale metode i projektet er: "Discount Method for Programming Language Evaluation". Metoden beskrives i et senere kapitel, men indtil videre konstateres det, at det i metoden er et mål at frigøre analysen af det undersøgte computersprog fra værktøjerne – altså editorer eller IDE. Adskillelsen sker 1) for at muliggøre at metoden kan anvendes i computersprogets designfase, altså inden det er færdigudviklet og implementeret i et udviklingsmiljø 2) fordi programmørerne måske ikke altid værdsætter eller helt forstår alle detaljer i forhold til sprogets syntaks.

Når de 14 dimensioner i Cognitive Dimensions bruges til at belyse et computersprog, er der nogle dimensioner, som næsten pr. automatik leder tankerne hen på det omkringliggende IDE i stedet for sproget selv. Det gør sig fx gældende for dimensionen "Secondary Notation". Dimensionen belyses blandt andet med spørgsmålet: "Is it possible to make notes to yourself, or express information that is not really recognised as a part of the notation?". Som det ses, spørges der ind til noget uden for det undersøgte computersprog.

Det er altså ikke sikkert, at en respondent i en konkret undersøgelse vil besvare alle 14 dimensioner, og at nogle derfor efterlades tomme. Det ses også i [9], hvor det flere steder i spørgeskemaet fremgår, at **hvis** respondenten svarer på dimensionen, bedes det uddybet med eksempler.

[7] "As a result, a number of programmers ... make comments that Cognitive Dimensions do not describe the programming language but the editor that one uses" [10].
[8] Se fx: [10], [11], [9] og [12].

7.4 Analyse af SAP HANA SQLScript

Nedenfor gennemføres en analyse af SAP HANA SQLScript. Analysen gennemføres ud fra Cognitive Dimensions og følger retningslinjerne i [10] og [11]. Det følger af problemformuleringen, at "Discount Metoden" undersøges i forhold til uddannelsen af programmører, og derfor anvendes Cognitive Dimensions også ud fra samme perspektiv, så der efterfølgende kan ske en sammenligning af resultaterne fra de to metoder.

Baggrunden er forfatterens egen tilegnelse af SAP HANA SQLScript. Det er således forfatterens egne erfaringer som programmør, der ligger til grund for undersøgelsen af sproget ved hjælp af Cognitive Dimensions. Forud for undersøgelsen har forfatteren gennemgået [13], [14] og [15]. Tilegnelsen af sproget omfattede ca. 50 timers arbejde i form af egne studier, internetbaseret undervisning og løsning af konkrete opgaver i SAP HANA SQLScript.

Når et computersprog undersøges ved hjælp af Cognitive Dimensions, kan det som tidligere beskrevet være svært at adskille sproget fra udviklingsmiljøet. Samtidigt ser der ud til at være en tendens til, at ikke erfarne programmører har problemer i forhold til notation. Det sidste fremgår fx af [2] "Empirical research indicates that inexperienced programmers struggle with notations". I undersøgelsen af SAP HANA SQLScript med Cognitive Dimension metoden, har det derfor været centralt ikke at bruge HANA Studio som hjælpeværktøj til at rette syntaksfejl. Syntakshjælp og brug af templates blev af den grund ikke anvendt, og alt kode blev gennemgået nøje, inden det blev forsøgt eksekveret i stedet for blot at forsøge at eksekvere koden, og herefter lade værktøjet HANA Studio finde og præsentere eventuelle syntaksfejl.

Efter forfatterens tilegnelse af HANA SQLScript blev relevante punkter i [9] udfyldt jf. vejledningen i [10] og [11].

Resultatet af udfyldelsen af spørgeskemaet ses i de de følgende underafsnit. En sammenfatning af resultaterne kan findes i tabel 10.1 på side 41.

7.4.1 Visibility

Dimensionen[9] bruges til at vurdere sprogets overskuelighed, og hermed også hvor gennemskuelige de forskellige dele af kodens funktioner er.

I linje 5 i listing 7.2 er det med `READS SQL DATA` angivet, at koden læser SQL data. Det ses også i linje 12 og 13, at koden rent faktisk læser data under brug af SQL. Det, der måske ikke er helt så åbenlyst, er at koden i linje 5 også betyder, at proceduren ikke skriver til eller opdaterer data i databasen. Det betyder også, at hvis der i funktionen kaldes øvrige funktioner, skal disse overholde de samme restriktioner.

Det fremgår heller ikke af linjen `READS SQL DATA`, at databasen som konsekvens vil forsøge at optimere koden internt til læsning af data.

Det må altså konstateres, at der er områder, hvor koden ikke er selvforklarende, men hvor yderligere viden er nødvendig for at kunne forstå virkningen af koden.

[9] Dimensionens betegnes nogle steder som Visibility & Juxtaposability.

7.4.2 Viscosity

Dimensionen bruges til at vurdere sprogets indbyggede modstand mod forandringer. Det kan fx afspejles i, hvordan en ændring et sted i et program medfører øvrige ændringer andre steder i programmet.

En Scalar Function kan måske ønskes ændret, efter den er defineret. Hvis en Scalar Function skal ændres, betyder det imidlertid, at den oprindelige function først skal slettes, og at en ny skal oprettes. Det kan opfattes som en simpel aktivitet, og det kan det også være i nogle tilfælde. Det betyder dog, at når en scalar function ændres, vil den først skulle slettes på udviklingsmiljøet, herefter på Q/A miljøet og til sidst på produktionsmiljøet. Bagefter kan den nye version af scalar funktionen oprettes på udviklingsmiljøet og transporteres til Q/A og herfra transporteres videre til produktionsmiljøet. Alt i alt vil det nok blive betragtet som en noget tung proces – særligt hvis der blot skal foretages en mindre ændring i funktionen.

Forholdet kunne også have været anført under dimensionen "Premature Commitment".

7.4.3 Error Proneness

Dimensionen bruges til at vurdere om sproget i sig selv lægger op til fejl eller misforståelser.

I SAP HANA SQLScript er det muligt at tildele en tabel til en variabel. En tabelvariabel kan enten erklæres eksplicit ved hjælp af DECLARE, eller den kan afledes "automatisk", når den tildeles en værdi. Det sidste har den fordel, at tabellens struktur ikke skal være kendt i forvejen.

Programmører kender måske begge muligheder for at oprette variable fra fx Microsoft Visual Basic for Applications.

Et eksempel på en eksplicit erklæret tabelvariabel ses i listing 7.2. I linje 7 ses, at der erklæres en variabel TTV (Temporary Table Variabel), der i linje 8 tildeles en tabel med én kolonne og én række med værdien 1[10].

```
1  CREATE PROCEDURE TV_EXPLICIT(OUT RESULTS TABLE (NUM INT))
2  LANGUAGE SQLSCRIPT
3  SQL SECURITY INVOKER
4  DEFAULT SCHEMA T4H
5  READS SQL DATA AS
6  BEGIN
7    DECLARE TTV TABLE(NUM INT);
8    TTV = SELECT 1 AS NUM FROM DUMMY;
9
10   BEGIN
11     DECLARE TTV TABLE(NUM INT);
12     TTV = SELECT 2 AS NUM FROM DUMMY;
```

[10]Dummy er en systemtabel i SAP HANA SQLScript, der altid indeholder én kolonne og én række. Dummy kan fx bruges til at tildele enkeltværdier til tabeller eller modtage resultater fra funktioner – i form af en tabel med én værdi.

```
13      RESULTS = SELECT * FROM :TTV;
14    END;
15
16    RESULTS = SELECT * FROM :TTV;
17  END;
18
19  CALL TV_EXPLICIT(?);
20  /* Resulterer i en tabel med en kolonne (NUM) og
21     en række med indholdet 1 */
```

Listing 7.2: Eksplicit erklæret variabler har globalt scope.

I linjerne 10-14 er der oprettet en selvstændig `BEGIN END` blok af kode. I koden erklæres TTV endnu en gang og tildeles nu værdien 2. I linje 13 tildeles procedurens output `RESULTS` værdien af `TTV`. Det samme sker i linje 16.

Når proceduren kaldes som fx i linje 19, viser det sig, at koden returnerer en tabel med én kolonne og én række, samt at denne celle indeholder værdien 1.

Det må således konkluderes, at variablen `TTV` kun har scope inden for den kodeblok, hvor den er erklæret. Med andre ord har variablen lokalt scope, når den erklæres eksplicit med `DECLARE`.

```
1   CREATE PROCEDURE TV_DERIVED(OUT RESULTS TABLE (NUM INT))
2   LANGUAGE SQLSCRIPT
3   SQL SECURITY INVOKER
4   DEFAULT SCHEMA T4H
5   READS SQL DATA AS
6   BEGIN
7   /*DECLARE TTV TABLE(NUM INT); */
8     TTV = SELECT 1 AS NUM FROM DUMMY;
9
10    BEGIN
11  /*  DECLARE TTV TABLE(NUM INT); */
12      TTV = SELECT 2 AS NUM FROM DUMMY;
13      RESULTS = SELECT * FROM :TTV;
14    END;
15
16    RESULTS = SELECT * FROM :TTV;
17  END;
18
19  CALL TV_DERIVED(?);
20  /* Resulterer i en tabel med en kolonne (NUM) og
21     en række med værdien 2 */
```

Listing 7.3: Afledte variabler har lokalt scope.

I listing 7.3 ses næsten den samme kode. Denne gang er linjerne 7 og 11 imidlertid kommenteret ud, så de ikke har effekt. Når koden eksekveres, opnås der et andet resultat end tidligere, idet koden returnerer en tabel med én kolonne og én række,

hvor cellen nu har værdien 2.

Det ses altså, at variablen oprettes, når den tildeles en værdi første gang, og at den samme variabel genbruges, når den tildeles en ny værdi, selvom det sker i en anden kodeblok. Med andre ord, så har afledte tabelvariable globalt scope.

Det snød ved flere lejligheder forfatteren, at måden, hvorpå en variabel oprettes, har betydning for dens scope. Det må antages, at jo længere brugeren bevæger sig væk fra variablens synlige oprettelse i koden, des mere sandsynligt er det, at der vil kunne forekomme misforståelser ved brug af variablen.

7.4.4 Closeness of Mapping

Dimensionen bruges til at vurdere, hvor nemt brugeren kan udtrykke sit mål.

SAP HANA SQLScript er en blanding af deklarativ SQL og imperativ programmering. I listing 7.4 er det vist, hvordan begge dele indgår i en procedure. I linjerne 12 til 20 ses typisk imperativ logik, hvor programmøren præcist angiver kodens forløb. I linje 22 og 23 ses derimod et typisk SQL udtryk med `SELECT FROM WHERE`.

```
1  /* Returner en tabel med alle dokumenter af en type
2     p.b.a. kald med blot et enkelt bogstav. */
3     CREATE PROCEDURE FILTER_BOOKS(IN CAT VARCHAR(1),
4                                   OUT RESULT TABLE(
5                                       BOOK_ID VARCHAR(3),
6                                       CATEGORY
                                           VARCHAR(10),
7                                       MRP DECIMAL(10, 2)))
8  LANGUAGE SQLSCRIPT
9  SQL SECURITY INVOKER
10 DEFAULT SCHEMA T4H
11 READS SQL DATA AS
12 BEGIN
13   DECLARE X VARCHAR(10);
14   IF :CAT = 'p' THEN
15     X := 'pdf';
16   ELSEIF :CAT = 'e' THEN
17   X := 'e-book';
18   ELSE
19     X := 'Printed';
20   END IF;
21
22   RESULT = SELECT BOOK_ID, CATEGORY, MRP FROM
23            "T4H"."Books" WHERE CATEGORY = :X;
24 END;
```

Listing 7.4: Eksempel på en SAP HANA SQLScript Procedure.

Det er imidlertid ikke altid, at de to former for kode kan blandes. I listing 7.5 ses en User Defined Function også kaldet en Scalar function i SAP HANA SQLScript. Selvom det i linje 4 angives, at koden er skrevet i SQLScript, er det i en User Defined Function kun muligt at anvende udtryk og den imperative del af SQLScript - ikke SQL. Den udkommenterede kode i 8 er således ikke lovlig.

Flere lignende tilfælde gav forfatteren problemer i forløbet.

```
1  /* Returner en pris fraregnet 20% rabat. */
2  CREATE FUNCTION FPRICE(IP DECIMAL(10, 2))
3  RETURNS RESULT DECIMAL(10, 2)
4  LANGUAGE SQLSCRIPT
5  SQL SECURITY INVOKER AS
6  BEGIN
7   RESULT := :IP - (:IP * (20/100));
8   /* RESULT := SELECT * FROM "Books";
9      Ulovligt kald af SQL */
10 END;
```

Listing 7.5: Eksempel på en SAP HANA SQLScript Function.

7.4.5 Hidden Dependencies

Dimensionen bruges til at vurdere, om der er skjulte afhængigheder indbygget i sproget, som programmøren selv skal tage højde for eller huske at vedligeholde.

Det forekommer som en åbenlys fordel, at programmøren selv i langt højere grad kan styre eksekvering af koden i forhold til brug af SQL. Det knap så åbenlyse er, at når programmøren vælger den imperative tilgang, træffer denne samtidigt et valg om, at afviklingen af kode sker sekventielt. HANA databasen kan således ikke optimere afviklingen af koden ved fx at bruge flere parallelle processer i flere af CPUens kerner.

Det er således op til programmøren selv at have tilstrækkelig indsigt i sproget og dermed kunne afgøre, hvornår kode med fordel kan skrives som "ren" SQL eller som imperativ kode i SAP HANA SQLScript.

7.4.6 Premature Commitment

Dimensionen bruges til at vurdere, om sproget tvinger programmøren til at træffe valg tidligt i processen, som har afgørende konsekvenser for den senere udvikling af funktionalitet.

Når en funktion (scaler) udvikles, skal der tages stilling til, hvilke rettigheder den skal udføres med i eksekveringstidspunktet. Det ses fx i listing 7.4 linje 9. Her skal programmøren altså tidligt tage stilling til, om funktionen skal indeholde `AS INVOKER` eller `AS DEFINER` (i dette tilfælde `AS INVOKER`). Altså om koden eksekveres med de samme rettigheder som brugeren, der kalder funktionen, eller med rettigheder som programmøren, der skriver funktionen.

7.4.7 Consistency

Dimensionen Consistency bruges til at vurdere den usikkerhed, der kan opstå, hvis den samme information kan udtrykkes på flere forskellige måder.

En kodeblok afsluttes med ordet `END` efterfulgt af semikolon. Forfatteren glemte i starten semikolon ved `END` i en del tilfælde, uden at dette havde betydning. Det viste sig senere, at årsagen skyldes, at `END` var den sidste linje i koden. I listing 7.4 ville semikolon således kunne udelades i linje 24 uden konsekvenser, mens en udeladelse af semikolon i listing 7.3 linje 17 vil give en fejl. Nogle vil måske opfatte denne eftergivenhed som en del af "Error Proneness", mens andre foretrækker at syntaksen er entydig. Her har erfaring og vaner fra tidligere anvendte sprog sandsynligvis også en indflydelse.

Når der skal tildeles en værdi til en variabel i SAP HANA SQLScript, gøres dette med `:=`. Denne tilgang kendes af mange fra fx sproget Pascal. Ved undersøgelse af en variabels værdi anvendes i stedet `=`.

Sondringen er kendt fra en række andre computersprog. For eksempel anvendes der i C++ et enkelt lighedstegn `=`, når en variabel tildeles en værdi, mens der anvendes dobbelt lighedstegn `==`, når indholdet af variablen undersøges.

I SAP HANA SQLScript kompliceres det yderligere af endnu en brug af kolon `:`. Når indholdet i en variabel skal anvendes i koden, foranstilles variabelnavnet med `:`. I forfatterens tilfælde var det årsag til adskillige fejl i de første timers introduktion til SQLScript.

Forholdet falder måske ikke ind under dimensionen "Consistency", men det var her forfatteren noterede det ved udfyldelsen af spørgeskemaet[9], hvori det fremgår "The questionnaire includes a series of questions that encourage you to think about the ways you need to use one particular notational system...". Det lægges derfor til grund, at det er respondentens tanker og oplevelser, der skal afdækkes, og at valget af dimension for en konkret oplevelse har mindre betydning.

7.4.8 Secondary Notation

Dimensionen bruges til at vurdere, om brugeren har mulighed for på en nem og uformel måde at tilføre eller tilgå informationer ud over det, der ligger i sprogets syntaks.

Forfatteren kunne uden problemer indføje kommentarer direkte i koden, ligesom koden kunne formateres ved brug af fx indrykning og indsættelse af blanke linjer.

7.4.9 Abstraction

Dimensionen bruges blandt andet til at vurdere, om sproget giver mulighed for at definere nye "facilities".

I listing 7.4 blev der vist en procedure, som har et enkelt bogstav som input, og som returnerer en tabel med tre kolonner som output.

Det fylder fire linjer at beskrive strukturen for output-tabellen (linjerne 4 til 7). Output-tabellen kunne i stedet beskrives som en Table Type, som kan genbruges i hele databasen i HANA.

Et eksempel på brug af Table Type kan ses i listing 7.6. I linje 1 til 3 oprettes en Table Type `TT_BOOK`. Denne indeholder kun strukturen for en tabel; ikke en faktisk tabel. `TT_BOOK` bruges herefter til at skabe proceduren i linje 5 og 6. Den definerede Table Type `TT_BOOK` kan herudover genbruges andre steder i databasen.

Denne mulighed for abstraktion lærte forfatteren hurtigt at sætte pris på.

```
1 CREATE TYPE TT_BOOK AS TABLE (BOOK_ID VARCHAR(3),
2                               CATEGORY VARCHAR(10),
3                               MRP DECIMAL(10, 2));
4
5 CREATE PROCEDURE FILTER_BOOKS(IN CAT VARCHAR(1),
6                               OUT RESULT TT_BOOK)
7 LANGUAGE SQLSCRIPT
8 ...
```

Listing 7.6: Brug af faciliteten Table Type.

7.5 Delkonklusion

Det er gennem kapitlet vist, hvordan Cognitive Dimensions kan anvendes til at vurdere og beskrive en notationsform – i dette tilfælde programmeringssproget SAP HANA SQLScript. Med Cognitive Dimensions kan man vælge at fokusere på det velfungerende i et sprog eller praktiske finesser, som det fx er beskrevet i forhold til brug af Table Types i afsnit 7.4.9. Da fokus med projektet er at afdække uddannelsesbehov, har det imidlertid været oplagt at bruge metoden til at finde og beskrive områder, hvor syntaksen kan drille, faldgruber og uklarheder ved sproget.

Igennem kapitlet er der beskrevet nogle forhold, som forfatteren enten ret hurtigt stødte på som hindringer, eller som opstod efter lidt mere arbejde med sproget, men som krævede yderligere indsats for at kunne forstå og anvende sproget.

Problemformuleringens andet underpunkt er altså besvaret, og der er ved brug af Cognitive Dimensions skabt et sammenligningsgrundlag for det efterfølgende forsøg og analyse med Discount Method for Programming Language Evaluation, jf. problemformuleringens tredje underpunkt.

8

Analysemodel

I dette kapitel følger en kort beskrivelse af "Discount Method for Programming Language Evaluation"[1], herunder metodens oprindelige formål, inden metoden efterfølgende anvendes i kapitel 9 til et andet formål end det, metoden oprindeligt var tiltænkt.

8.1 Discount Method for Programming Language Evaluation

Formålet med "Discount Method for Programming Language Evaluation" beskrives som: "The method is intended to bridge the gap between small scale internal language design evaluation methods and large scale surveyes and quantitative evaluation methods. The method is designed to be applicable before a compiler or IDE is developed for a new language".

Som det ses af citatet, inddrager metoden ressourcer, som ikke er blandt udviklerne i forhold til design af det nye sprog – uden at det må eskalere til en større undersøgelse.

Metoden, der er under fortsat udvikling ved Aalborg Universitet, omfatter i alt 9 aktiviteter, der kort beskrives nedenfor.

8.1.1 De 9 aktiviteter

1. **Create tasks.** Disse opgaver er specifikke i forhold til det computersprog, der undersøges, og skal afdække sprogets nøglefunktioner. Det kan være en brugbar tilgang at udarbejde scenarier, som det forventes, at brugeren skal anvende sproget til, og som dækker det, brugeren har behov for i relation til at kunne løse sine opgaver.

[1]Dette afsnit er baseret på [1]. For hver af metodens 9 punkter er overskriftens engelske navn fastholdt, mens essensen af forklaringerne er forkortet og frit oversat til dansk.

2. **Create a Sample Sheet.** Efter at have defineret opgaverne i pkt. 1, er der nu en idé om, hvad brugeren skal vide for at kunne løse opgaverne, og der kan nu udarbejdes et eksempelark, eller Cheat Sheet, som det kendes fra mange computersprog. Små stumper eksekverbar kode kan give brugeren en bedre forståelse af sprogets overordnede struktur.

3. **Estimate the task duration.** Ved at tage tid på egen løsning af opgaverne kan der fås en idé om, hvor lang tid deltagerne i det efterfølgende forsøg vil være om at løse opgaverne. Forsøgsdeltagerne vil sandsynligvis være længere tid om at løse opgaverne, idet de først skal vænne sig til sprogets struktur og syntaks. I metoden anbefales det, at have flere opgaver end det forventes, at forsøgets deltagere vil kunne løse. Det anbefales, at de vigtigste opgaver placeres først, og at deltagerne gøres opmærksom på, at de ikke forventes at kunne løse alle opgaverne.

4. **Prepare setup.** Den konkrete opsætning kan variere mellem et komplet udviklingsmiljø til brug af papir og blyant. I metoden anbefales det, at forsøgene optages for bedre efterfølgende at kunne evaluere løsningen af opgaverne. Klargøring af udstyr indgår også som en aktivitet under dette punkt.

   Conduct a pilot test. Fejl eller mangler i opgaverne eller eksempelarket kan rettes forud for de egentlige forsøg ved først at gennemføre en test.

5. **Gather participants.** Metoden foreskriver, at fem forsøgsdeltagere er det optimale. Hvis der anvendes mere end fem deltagere, kan det give dublerede observationer (der dog kan underbygge observationen). Hvis der anvendes færre end fem deltagere, kan der være tendens til, at væsentlige problemer ikke afdækkes.

6. **Start the experiment.** Under dette punkt er det vigtigt, at deltagerne gøres klart, at det er computersproget, der afprøves – ikke deltagerne. Formålet med punktet er at imødegå en unødvendig nervøsitet blandt deltagerne.

7. **Keep the participants talking.** Under forsøget skal facilitator tilstræbe, at deltagerne løbende fortæller, hvad de tænker om løsningen af den opgave, de er i gang med. Facilitatoren må gerne besvare spørgsmål og drøfte løsninger med deltagerne. Det skyldes, at forsøget ikke drejer sig om at teste deltagernes evne til at formulere løsninger, men om hvordan deltagerne omsætter deres løsninger til kode. Da "systemet" ikke selv giver feedback, skal facilitatoren oplyse, hvornår en opgave er løst.

8. **Interview the participants.** Når forsøget er afsluttet, gennemføres der et kort interview med deltageren. Her kan facilitatoren drøfte både det anvendte computersprog, opgaverne og eksempelarket. Modellen anbefaler, at der til interviewet udarbejdes et lille spørgeark til facilitatoren.

9. **Analyser data.** Efter forsøgene er gennemført, anvendes de opsamlede data til at opstille en liste med de problemer, der er blevet afdækket. Problemerne kan inddeles i følgende kategorier:

- **Kosmetiske problemer** der omfatter mindre slåfejl og lignede, der nemt vil kunne rettes.
- **Seriøse problemer** der omfatter strukturelle fejl, der kræver ændringer i kodens struktur, men som alligevel er så små, at de vil kunne rettes med få ændringer.
- **Kritiske problemer** der omfatter fundamentale misforståelser af, hvordan kode struktureres i sproget og større fejl, som kræver gennemgribende ændringer i koden.

Resultatet af de kategoriserede problemer er en prioriteret liste af forbedringsmuligheder til det afprøvede computersprog.

9 Undersøgelse

I forbindelse med projektet er der gennemført en undersøgelse jf. processen beskrevet i afsnit 8.1.1 med det formål at afdække, om Discount Metoden kan anvendes til at vurdere uddannelsesbehovet ved respondenterne ved omstilling til SAP HANA SQLScript. Der har været anvendt i alt syv respondenter til brug for den samlede undersøgelse. Én er brugt som pilot for at afprøve Cheat Sheet og Task Sheet, mens de øvrige seks har været egentlige respondenter i undersøgelsen[1]. De seks respondenter udgøres af de IBM konsulenter, der p.t. arbejder med vedligeholdelse og udvikling af Forsvarets BI løsninger på det kodenære niveau. Respondenterne spænder fra en juniorkonsulent med ca. to års erfaring til en specialistkonsulent med ca. 30 års erfaring. På det programmeringsfaglige område har flere meget bred erfaring med programmering i adskillige computersprog, mens enkelte "blot" har kendskab til SQL eller fejlfinding i implementering af kode, der er udviklet af andre programmører.

Respondenten, der er anvendt som pilot, er en af Forsvarets ressourcepersoner, der almindeligvis gennemfører kvalitetskontrol på de øvrige respondenters produkter.

9.1 Klargøring

Task Sheet og Sample Sheet (der også er kendt som Cheat Sheet) er to centrale elementer i Discount Metoden. Herudover ligger det implicit i metoden, at indsatsen fra de involverede skal begrænses. Det er derfor oplagt at gøre en stor indsats i forbindelse med klargøringen, så den efterfølgende gennemførelse af forsøgene kan bidrage med flest mulige anvendelige data med lavest mulig indsats.

[1] Følgende fremgår af [1] "The golden rule for the number of praticipants is five. More than that and most of the encountered problems are ones you already have observed...". På tidspunktet for undersøgelsen bestod IBMs Business Intelligence team af seks programmører, der alle deltog, idet det af andre hensyn ikke var et ønske om at fravælge en enkelt.

9.1.1 Cheat Sheet

Det har ikke været muligt at finde eksisterende Cheat Sheet for SAP HANA SQLScript. Der er derfor udarbejdet et to siders Cheat Sheet til forsøgene. Cheat Sheet kan findes i Bilag C.

Et Cheat Sheet kan ikke rumme et sprogs komplette syntaks eller eksempler på den fulde anvendelse af sproget. Der er under udarbejdelsen derfor lagt vægt på de mest anvendte konstruktioner og særegne egenskaber, herunder kombinationen af deklarativ og imperativ programmering:

- I punkt 4. vises brugen af en Scalar Function, der kan indeholde imperative udtryk.
- I punkt 6. vises brugen af en Table Function. Denne defineres "design time" og kan derfor transporteres mellem systemer fx igennem et typisk flow fra et development miljø, til et test miljø og til sidst et produktionsmiljø.
- I punkt 7. ses, hvordan Stored Procedures kan anvendes i databasen til at ændre indholdet af en tabel.

De tre fremhævede områder ovenfor er særegne for SAP HANA SQLScript. Det kunne overvejes, om puntk 7. om Stored Procedures burde fremtræde før punkt 6. vedrørende User Defined Table Functions. Det konkrete valg er truffet i forhold til, hvordan respondenterne i det konkrete forsøg vil skulle bruge oplysningerne.

9.1.2 Opgaver

Til forsøgene skal der også anvendes et opgaveark, som fordrer brug af nogle af sprogets særegne kendetegn, eller er målrettet mod brugerens kommende opgaver under anvendelse af sproget.

Til det brug er der udarbejdet et opgavesæt med i alt fem opgaver. Opgaverne kan forekomme ret banale, men de tager udgangspunkt i virkeligheden for en typisk BI programmør i Forsvaret. Opgaverne findes i bilag A og et sæt mulige løsninger findes i bilag B.

Opgaverne er designet, så der er en stigende progression i sværhedsgraden. Herudover er det tilstræbt at dække de forskellige problemstillinger, som en programmør typisk vil skulle gennemgå i sit første bekendtskab med SAP HANA SQLScript.

9.1.3 Undersøgelsens praktiske forhold

9.1.3.1 "Spilleregler"

Forud for undersøgelsens start var de seks respondenter og personen til pilotafprøvningen samlet til en kort introduktion til undersøgelsen, herunder baggrunden for undersøgelsen og formålet med undersøgelsen. De fik her forklaret det egenlige formål med Discount Metoden samt, at det nu skulle undersøges, om metoden kan bruges til andre formål fx at fastlægge uddannelsesbehov. De fik ikke at vide, at SAP HANA SQLScript ville indgå i undersøgelsen. Endeligt blev de praktiske forhold gennemgået:

1. **Tidsforbrug.** Deltagerne blev orienteret om, at forsøget ville tage 60–90 minutter for hver af dem. Forsøget ville blive gennemført i arbejdstiden, og at deres respektiver chefer havde godkendt deres deltagelse i forsøget. Der var stor nysgerrighed i forhold til undersøgelsen, og alle var meget positivt indstillet i forhold til at deltage.

2. **Cheat Sheet og opgavesæt.** Deltagerne blev orienteret om, at de i starten af forsøget ville få et Cheat Sheet, som de ville få tid til at danne sig et overblik over, og at de efterfølgende ville få et sæt opgaver, de skulle løse. Det blev flere gange understreget, at de ikke ville kunne nå at løse alle opgaverne, og at det heller ikke var et mål i sig selv. De blev i stedet opfordret til at "tænke højt" under forsøget og have en dialog med observatøren.

3. **Værktøjer.** Det blev beskrevet, at "udviklingsværktøjet" var Microsoft WordPad. Det krævede lidt forklaring, herunder en gentagelse af at Discount Metoden er udviklet til at afprøve nye computersprog, før de er færdigudviklet og muligvis før, der er et udviklingsmiljø klart. Det blev også drøftet, at den manglede syntakshjælp m.m. i WordPad kan være med til at afdække flest mulige uhensigtsmæssigheder ved et sprog.

   Deltagerne blev også orienteret om, at programmet Snagit ville være aktiveret under forsøget, således at alt skærmaktivitet og tale i lokalet ville blive optaget.

4. **Afslutning.** Deltagerne fik oplyst, at forsøget ville blive afsluttet med et kort interview om, hvad der var let og svært i forbindelse med løsning af opgaverne, deres oplevelse af computersproget samt lidt om deres baggrund – primært erfaring med forskellige computersprog.

5. **Generelt.** Det blev flere gange under introduktionen fremhævet, at det ikke var respondenterne, der ville blive undersøgt under forsøget, men derimod "Discount Metoden", idet dennes anvendelighed til at estimere uddannelsesbehov ved skift af programmeringssprog skulle vurderes – altså om metoden kan anvendes til andet end det, den oprindeligt er udviklet til.
   Endeligt blev deltagerne gjort bekendt med, at data behandles efter Forsvarets bestemmelser om Persondata (GDPR), herunder at persondata slettes, når de ikke længere har relevans.

9.1.3.2 Dataindsamling

Det fremgår implicit af navnet "Discount Metode", at metoden kræver lille indsats for at kunne levere brugbare resultater. Det er således oplagt, at forsøget skulle planlægges, så flest mulige relevante og tilgængelige data indsamles under forsøget og tilbageløb undgås.

- Koden som respondenterne udarbejdede under forsøget skulle gemmes umiddelbart efter forsøget var afsluttet. Det var imidlertid klart, at den "færdige" kode, kun ville vise slutproduktet, altså ikke hvordan den blev udviklet, herunder hvilke rettelser, der blev foretaget undervejs eller hvilke områder, der fx var tidskrævende. Koden blev planlagt gemt i MS WordPad *.docx format.

- Al skærmaktivitet og lyd i forsøgslokalet (samtale mellem respondenten og observatøren) blev optaget af programmet Snagit. Data var planlagt gemt i mp4 format umiddelbart efter forsøgtes afslutning. Under planlægningen blev det overvejet, om der skulle optages levende billeder udover aktiviteterne på skærmen, således at fx respondenternes kropssprog også blev optaget. Det blev fravalgt af praktiske årsager.[2]

- Mikrofonen i den computer, der blev anvendt under forsøget, er i nogen grad retningsbestemt, og den kan derfor have svært ved at opfange samtale i rummet. Lyden blev derfor også optaget på en mobiltelefon som backup.

- For at lette den senere behandling af de øvrige data, tog observatøren notater undervejs. Notaterne refererede til en specifik opgave i opgavesættet eller henviste til et klokkeslæt. I Microsoft Windows på forsøgscomputeren blev "Skjul automatisk proceslinje i skrivebordstilstand" slået fra, således at uret i nederste højre hjørne hele tiden var synligt, og det blev optaget som en del af det samlede skærmbillede af Snagit. På den måde var det nemt at referere til et tidspunkt, der senere kunne genfindes i optagelserne.

9.2 Gennemførelse af undersøgelsen

Undersøgelserne med respondenterne blev i det store hele gennemført som planlagt og beskrevet ovenfor. Det var intentionen at anvende 60 minutter effektivt til løsning af opgaverne. Herudover var der afsat tid til både indledning og et kort efterfølgende interview, der skulle afdække, hvad der var let og svært for respondenterne, samt afdække deres erfaringer inden for området. Den samlede udstrækning af et forsøg skulle således kunne holdes inden for 90 minutter.

Det viste sig allerede i forbindelse med pilotafprøvningen og det første forsøg, at respondenterne blev engagerede i løsningen af opgaverne. Det medførte i praksis, at hvert forsøg blev afbrudt ved afslutningen af en opgave efter de 60 minutter, således at respondenterne ikke blev afbrudt midt i løsningen af en opgave.

9.3 Analyse af undersøgelsen

Punkt nummer 9 i "Discount Method for Programming Language Evaluation" foreskriver, at data analyseres med henblik på at opstille en liste med problemer, der afdækkes under forsøget, og at problemerne kategoriseres.

En grundlæggende tanke bag metoden er, at den skal være enkel. Det er således ikke yderligere beskrevet, hvordan analysen gennemføres, og kategoriseringen omfatter "blot" en inddeling af problemerne i tre kategorier jf. afsnit 8.1.1, hvor kategoriseringen sker ud fra en vurdering af, hvor stor en indsats det vil kræve at rette fejlen.

Undersøgelsens resultater findes som kommentarer i transskriberingerne af forsøgene vedlagt som bilag D. Den resterende del af dette kapitel sammenfatter analysens

[2]Det ville kræve en særlig sikkerhedsgodkendelse at filme i de bygninger, hvor forsøget gennemførtes.

resultater for hver af de seks respondenter benævnt R1 til R6. En sammenfatning af resultaterne kan findes i tabel 10.1 på side 41.

9.3.1 Respondent 1

Opgave 1

- R1 foretager sikkert valg af `DECIMAL` som input og output.
- R1 definerer variable med `in_` og `out_` prefiks.
- R1 er muligvis i tvivl om brugen af forkortelserne `DEC` og `INT` for hhv. `DECIMAL` og `INTEGER`.
- R1 Følger eksemplet i Cheat Sheet slavisk.

Opgave 2

- R1 er lidt usikker på adskillelse af inputparametre – skal der anvendes semikolon?
- R1 gennemlæser koden og foretager rettelser sikkert – støttet af Cheat Sheet.

Opgave 3

- R1 er sikker i gennemførelsen af opgaven, herunder implementering af logikken.

Opgave 4

- R1 arbejder koncentreret og anvender i høj grad Cheat Sheet.
- Flere rettelser i koden indikerer, at R1 er usikker på brugen af Schema og Repository.
- R1 accepterer linjen `SQL SECURITY AS INVOKER` som noget, der bare skal stå der.

Opgave 5

- R1 er sikker i valget af en procedure.
- R1 er usikker på beskrivelse af inputparametre til proceduren.
- R1 er usikker på scope for tabellen.
- R1 er usikker på tilføjelse af data til tabellen.
- R1 bruger noget tid på logikken i proceduren. Det er næppe sprogafhængigt.
- R1 er usikker på hvordan en værdi castes fra `DECIMAL` til `INTEGER`, men han er bevist om, at der kan være et behov.

Respondenten R1 er en erfaren programmør. R1 har cirka et år forud for forsøgets gennemførelse læst kursusmaterialet til SAP HANA150, men R1 har ikke gennemført kurset. Under løsningen af opgaverne opstår der i mindre grad kosmetiske fejl. Den egentlige logik i opgaverne løses sikkert. R1 udviser usikkerhed i forbindelse med den underliggende sikkerhedsmodel i HANA samt scope for tabeller m.m. I de konkrete opgaver giver det ikke anledning til problemer, men det må antages, at den manglende forståelse vil kunne føre til problemer i kategorien seriøse problemer.

9.3.2 Respondent 2

Opgave 1

- R2 forekommer lidt nervøs og usikker i situationen.

- Facilitator hjælper og vejleder R2 lidt mere end hos de øvrige respondenter for at sikre en god start og prøve at fjerne den indledende nervøsitet.
- R2 bruger undervejs punktum efter `END`. R2 sletter det efterfølgende.

Opgave 2
- R2 er usikker i forhold til, hvordan logikken kan implementeres. Det indikerer grundlæggende udfordringer i forhold til imperativ programmering.
- R2 er usikker i forhold til at skrive en rutine, som kan behandle to inputparametre.
- R2 er usikker i forhold til, hvad en `INTEGER` kan indeholde.
- R2 forsøger at finde "mønstre" i koden i stedet for at klarlægge syntaks.
- R2 er noget usikker på forgrening ved hjælp af `IF-ELSEIF-ELSE-ENDIF`.

Opgave 3
- Facilitator vejleder og støtter noget mere end hos de øvrige respondenter.
- R2 ser ikke ud til at have tilstrækkelig grundlæggende viden om imperativ programmering til at kunne opstille logikken, der er nødvendig for at kunne løse opgaven.
- R2's forklaring på forskellen mellem `=` og `:=` er usammenhængende og giver ikke entydig mening.

Opgave 4
- Facilitator vejleder og støtter noget mere end hos de øvrige respondenter.
- R2 er hurtig til at pege på SQL. Det indikerer, at R2 er mere rutineret i brug af SQL sammenlignet med imperativ programmering.
- R2 er usikker i forståelse af HANA database teknologi, herunder forståelse for brugen af Schema og Repository.
- R2 støtter sig i høj grad til koden i Cheat Sheet.

Respondenten er en trainee med begrænset erfaring. Både løsning af opgaverne og det korte interview efterfølgende viser, at der er et vist kendskab til SQL, mens erfaringerne inden for imperativ programmering er meget begrænsede. De afdækkede fejl i koden omfatter kategorierne: kosmetiske og seriøse. Da opgavernes omfang er beskedne, vil det næppe være muligt at finde fejl i kategorien kritisk. Det er dog nærliggende at antage, at respondenten uden yderligere uddannelse ville kunne skabe kritiske fejl ved løsning af større opgaver, herunder særligt fejl i forbindelse med den imperative del af SAP HANA SQLScript.

9.3.3 Respondent 3

Opgave 1:
- R3 forstår ikke definitionen af `DECIMAL(x,y)` og tror fejlagtigt, at `x` er størsteværdien – ikke antal cifre. R3 definerer derfor `PUND DECIMAL(1000,2)`. Efter kort drøftelse er R3 helt bevidst om fejlen, og hvordan syntaksen er.

Fejlen falder i kategorien kosmetiske problemer.

Opgave 2:

- R3 anvender `IF-ELSEIF-ELSE-ENDIF` lidt omstændigt. Koden er dog eksekverbar.
- Der er fejl i antallet af parentes slut i forbindelse med definition af `DECIMAL`.
- R3 medtager ikke `:`, når der refereres til indholdet af variablen.

Første observation er ikke en fejl. Anden og tredje observation er kosmetiske fejl.

Opgave 3:

- R3 har problemer med `:` foran variabelnavne, når der refereres til indholdet.
- R3 undlader `;` efter `END`.

Samlet set spores der lidt usikkerhed i forhold til den imperative implementering af logikken. Løsningen resulterer imidlertid i eksekverbar kode, og problemerne kategoriseres som kosmetiske.

Opgave 4:

- R3 er usikker på, hvordan funktionen navngives.
- R3 er usikker på kald med sikkerhedsrettigheder som `SQL SECURITY INVOKER`.

Første observation vurderes at være kosmetisk. Anden observation giver ikke anledning til konkrete problemer i opgaven. Manglende kendskab til den underliggende sikkerhedsmodel i HANA og dermed brugen i SAP HANA SQLScript kan give anledning til seriøse problemer.

Samlet set konkluderes det, at Respondent 3 er en erfaren programmør, som behersker både SQL og imperativ programmering. Løsning af opgaverne indikerer dog, at rutinen i imperativ programmering er lidt "rusten", mens SQL er mere flydende. De observerede fejl er fortrinsvis i kategorien kosmetiske fejl, hvor det må antages, at brug af korrekt syntaks hurtigt vil kunne indøves.

9.3.4 Respondent 4

Opgave 1

- R4 resonerer godt vedr. valg af variable, både input og output.
- R4 bemærker, at han nok vil gå tilbage til antallet af decimaler i variablerne.
- R4 er så optaget af Cheat Sheet, at han glemmer logikken i rutinen.

Opgave 2

- R4 virker lidt usikker på definitionen af en `DECIMAL`, i hvert fald som output til en timeløn.
- R4 resonerer fint i forhold til brug af `IF-ELSIF-ENDIF`.
- R4 er usikker i brugen af `:` til at referere en variabel.
- R4 er usikker på forskellen mellem `:` og `:=`.

Opgave 3

- R4 har fin forståelse for variabel- og objecttyper.
- R4 starter med at skitsere loop med papir og blyant.
- R4 er usikker vedrørende reference til variable, herunder brugen af `:`.
- R4 anvender ikke `:=` ved tildeling af en værdi til en variabel.

Opgave 4

- Facilitator støtter R4 for at komme i gang med opgaven.
- R4 er usikker i brugen af Schema og Repository.
- R4 er usikker på, hvad det vil sige at returnere en "table".
- R4 efterlyser en navnestandard. R4 retter derfor navne på variable og funktion til mere klare navne.
- R4 virker mere sikker i SQL delen af opgaven – sammenlignet med brugen af imperativ programmering i de foregående opgaver.
- R4 er lidt usikker på forskellen imellem variabelnavne og kolonnenavne. R4 erkender dog forskellen i løbet af opgavens løsning.

Der er tale om en rutineret programmør. Koden fra R4 er undervejs præget af en række kosmetiske fejl, herunder brugen af `=`, `:=` og `;`. R4 forsøger sig fx med flere tilgange til løsning af opgave 3, før han lægger sig fast på en tilgang. Som den eneste af respondenterne anvender R4 et andet værktøj til løsning af opgaven, idet han kort skitserer loop i opgave 3 med papir og blyant undervejs i løsningen.

R4 er generelt noget usikker i forhold til at implementere logik i imperativ kode, og R4 forklarer flere gange undervejs, at han er mest vant til at lade sig inspirere af eksisterende kode.

R4 introducerer ikke seriøse problemer, men særligt i forbindelse med løsning af opgave 4, er der en usikkerhed, som måske ville kunne lede til seriøse eller kritiske problemer.

9.3.5 Respondent 5

Opgave 1

- R5 er usikker på definitionen af `DECIMAL(t, f)`.
- R5 glemmer `;` efter `END`.

Opgave 2

- R5 kommer til at bruge division i stedet for gange. Fejlen vurderes ikke at være afhængig af sproget.
- R5 er usikker på brugen af `:` foran variable, men resonerer sig frem til betydningen.
- R5 er usikker på forskellen på `=` og `:=`. R5 mener, at `=` burde være nok i begge situationer, idet konteksten forklarer betydningen.
- R5 glemmer `;` efter `END`.

Opgave 3

- R5 Mangler `BEGIN END` i sin kode.
- Logikken i funktionen driller R5. Det indikerer, at R5 er lidt usikker i imperativ programmering.
- R5 glemmer flere gange `:` foran variabler, der refereres.
- R5 bruger `:` foran variabel, hvor der ikke refereres.

Opgave 4

- R5 vælger ikke at kopiere tidligere kode men at skrive det hele - meget støttet af Cheat Sheet. På den måde har R5 tid til at tænke undervejs.
- R5 er lidt usikker på brugen af Schema og Repository.

R5 er en rutineret og eftertænksom programmør. Der forekommer kosmetiske fejl i koden fra R5. Når R5 følger Cheat Sheet, sker det i et roligt men sikkert tempo, hvor han undervejs forholder sig til kodens funktion og sætter denne i relation til eget behov for løsning af en opgave. I løsningen af opgave 4 vælger R5 meget bevidst ikke at kopiere tidligere kode men i stedet skrive alt fra bunden og holde hovedet koldt undervejs.

R5 udviser nogen usikkerhed i forhold til imperativ programmering. Det ses mest udpræget i opgave 3. Herudover er R5 noget usikker i relation til opbygningen af en HANA database, herunder brugen af Schema og Repository.

9.3.6 Respondent 6

Opgave 1

- R6 følger eksemplet i Cheat Sheet tæt.
- R6 definerer input som `INTEGER`. Adspurgt er R6 helt klar over konsekvensen.

Opgave 2

- Facilitator hjælper R6 for at komme godt i gang med opgaven.
- R6 er usikker på brugen af datatyper. R6 bruger `INTEGER` som output fra funktionen, og erklærer `INTEGER` med parametrene `(3,0)`.
- R6 følger eksemplet i Cheat Sheet meget tæt.
- R6 er usikker på bruger af `DEC(t, f)`.
- R6 er usikker på forskellen mellem `=` og `:=`.

Opgave 3

- R6 definerer `INTEGER(9,0)`. R6 er muligvis ikke klar over, at der ikke er decimaler på en `INTEGER`.
- R6 har problemer med at omsætte logikken til imperativ kode. R6 synes at have de "rigtige" tanker, men R6 har svært ved at omsætte dem til imperativ kode.
- Opgaven ender med ikke sammenhængende kode.

Opgave 4

- R6 ser ikke helt ud til at have forstået det "smarte" ved at kunne kalde en funktion med parametre, der efterfølgende anvendes i SQL kaldet.
- R6 fokuserer på at få koden i overensstemmelse med Cheat Sheet snarere end at tænke over, hvad der rent faktisk skal ske i koden.

R6 fortæller, at han i sit daglige arbejde igennem en del år mest har læst andres kode – ikke selv skrevet kode. Det fremgår også ret tydeligt under forsøget. I de første to opgaver er det i høj grad muligt at støtte sig til Cheat Sheet, hvilket R6 også gør. I de efterfølgende opgaver går det ikke helt så nemt. R6 "tænker højt" under løsning af opgaverne, og han har tydeligvis de rigtige tanker, men det kniber med at omsætte dem til egenligt eksekverbar kode. R6 er mest fokuseret på at følge eksemplerne i Cheat Sheet og fokuserer ikke på, hvilken opgave koden rent faktisk skal løse. Særligt opgave 3 fører til problemer i kategorien seriøs.

Der er kosmetiske fejl, men da R6 følger Cheat Sheet tæt, er der relativt få, og koden fremstår velordnet.

10 Konklusion

I dette kapitel sammenlignes resultaterne af analysen i kapitel 7, hvor Cognitive Dimensions blev anvendt på sproget SAP HANA SQLScript, med resultaterne i kapitel 9, hvor Discount Method for Programming Language Evaluation i stedet blev anvendt.

Herefter beskrives det, hvad brugen af Discount Metoden har afdækket i forhold til Forsvarets aktuelle forhold, når de nuværende programmører skal omstilles til SAP HANA SQLScript.

Endeligt samles der op på delkonklusioner fra behandlingen af de øvrige underpunkter i problemformuleringen, således at den samlende konklusion fremstår som et hele.

10.1 Sammenligning af undersøgelsens resultater med resultaterne fra Cognitive Dimensions.

Essensen af resultaterne fra analysen i kapitel 7, hvor Cognitive Dimensions blev anvendt og kapitel 9, hvor Discount Metoden blev anvendt er sammenfattet i tabel 10.1. Den øverste tredjedel af tabellen viser de områder, hvor de to metoder begge afdækkede de samme forhold. Den efterfølgende tredjedel af tabellen viser de områder, der blev afdækket af Cognitive Dimensions – men ikke med Discount Metoden i den konkrete case. Den sidste del af tabellen viser de områder, som blev afdækket af Discount Metoden, men som ikke blev identificeret med Cognitive Dimensions.

Som det fremgår af den øverste tredjedel af tabellen, så er der i alt fem områder, som blev fundet ved brug af både Cognitive Dimensions og Discount Metoden. De første tre forhold er tæt knyttet til ofte anvendt syntaks i SAP HANA SQLScript. Det ses dog også i de to resterende fælles områder, vedrørende brug af Schema og Repository, samt sikkerhedsmodellen i HANA, at begge modeller når dybere end

det, der ligger i overfladen i form af hyppig anvendt syntaks.

Den midterste del af tabel 10.1 viser de områder, der blev afdækket med Cognitive Dimensions, men ikke rigtigt kom frem i forsøget og analysen med Discount Metoden. Der er tale om mere komplekse dele af SAP HANA SQLScript, som kun i begrænset omfang er dækket af Cheat Sheet – i pkt. 6 og 7.

Det var kun få af respondenterne, der nåede til opgaver, der berører disse områder. Det har ikke været muligt sikkert at efterprøve respondenternes forståelse for de komplekse områder inden for de afgrænsninger, der ligger i Discount Metoden. Når de komplekse dele anvendes i praksis, skal der udarbejdes en del kode, hvor udviklingsværktøjet (HANA Studio) typisk vil kunne autogenerere større dele. Praktisk undersøgelse af programmørernes parathed har derfor ikke kunne undersøges i MS WordPad – og ikke inden for den tidsramme, der ligger i Discount Metoden.

Her kommer Discount Metoden måske til kort som følge af både tids- og værktøjsbegrænsningerne.

Endeligt har forsøget med Discount Metoden afdækket nogle forhold, som ikke blev erkendt med Cognitive Dimensions, herunder problemer med forståelse af datatyper og deres definition samt generelle problemer med at omsætte opgaveløsning til imperativ programmering. At disse forhold ikke blev afdækket af Cognitive Dimensions kan næppe tilskrives begrænsninger i metoden. Det er nok snarere et udtryk for, at flere af respondenterne ikke har de samme forudsætninger inden for imperativ programmering som forfatteren.

I forhold til problemformuleringens tredje underpunkt i kapitel. 5 kan det konkluderes, at Discount Metoden i den konkrete case har givet resultater, der svarer til de resultater, der blev opnået med Cognitive Dimensions – ikke mindst når der sammenlignes på de områder, hvor fejl eller uklarheder tidligt bliver tydelige, når et nyt sprog introduceres.

10.2 Afdækning af Forsvarets aktuelle forhold.

I forhold til den undersøgte case, så har analysen ved hjælp af Discount Metoden vist, at der er en nogle gennemgående kosmetiske fejl. Det drejer sig primært om at afslutte logisk afgrænsede blokke af kode med semikolon.

Hertil kommer særligt brug af kolon. Problemerne opstår både i forbindelse med sondring mellem tildeling af værdi til en variabel ved brug af kolon lig med `:=` og ved undersøgelse af en variabels værdi, hvor der blot anvendes lig med `=`. Det er også gennemgående, at kolon glemmes, når der refereres til indholdet af en variabel. Det vurderes dog, at forskellen hurtigt vil komme til at "ligge i fingrene", og at det bliver en vane at skrive det rigtigt. Denne hyppige fejl indgår derfor også i kategorien af kosmetiske fejl.

Det næste område drejer sig om imperativ programmering. Det står klart, at der er stor forskel i respondenternes parathed på området. R1 og R3 ses at være helt fortrolige på området, og det vurderes, at det er ikke nødvendigt med yderligere målrettet uddannelse til de to. For de øvrige og ikke mindst R2 og R6 er det nød-

	CDs	R1	R2	R3	R4	R5	R6
Problem							
Semikolon efter END	X[a]			3		2	
Lig med (=) og kolon lig med (:=)	X[a]		3		2+3		2
Reference til en variabels indhold (:VAR)	X[a]			2+3	2+3	3	
Schema og Repository (kan koden tranporteres, samt syntaks)	X[b]	4+5	4		4	4	
Sikkerhedsmodel (SQL SECURITY AS INVOKER)	X[c]	4		4			
Variblers scope lokal eller global	X[d]						
"Fravalg" af automatisk optimering af kode	X[e]						
Fordel ved brug af Table Type	X[f]						
Datatyper fx INTEGER og DECIMAL			2		2		2
Imperativ programmering			2		4	3	3
Andet værktøj (papir og blyant)					3		

[a] Jf. afsnit 7.4.7.
[b] Jf. afsnit 7.4.2.
[c] Jf. afsnit 7.4.6.
[d] Jf. afsnit 7.4.3.
[e] Jf. afsnit 7.4.5.
[f] Jf. afsnit 7.4.9.

Tabel 10.1: Sammenligning af resultater fra Cognitive Dimensions og Discount Method for Programming Language Evaluation.

vendigt at gennemføre en målrettet uddannelse i imperativ programmering, før disse vil kunne bidrage til leverancer til Forsvaret, hvor brug af SAP HANA SQLScript indgår.

Flere af de respondenter, der mangler færdigheder inden for imperativ programmering, ses i de konkrete opgaver at have de grundlæggende tanker på plads, men det kniber med at omsætte tankerne til eksekverbar kode. Det peger på, at det ud over en introduktion til imperativ programmering eller genopfriskning af samme også er nødvendigt med noget træning fx gennem løsning af opgaver. Der er altså en gruppe af programmører, som har behov for yderligere uddannelse ud over "blot" indføring i en ny syntaks.

For at kunne udnytte mulighederne i SAP HANA SQLScript, er der i høj grad behov for, at programmøren kan tænke på databaseniveau og ikke kun på applikationsniveau. Dette er forsøgt afdækket med de sidste opgaver i opgavearket.

I praksis er det dog svært at afgøre, i hvilken udstrækning respondenterne reelt har forstået forskellen på at kode til databasen og til applikationsserveren, eller om de "blot" formår at adoptere og tilpasse koden fra Cheat Sheet. I forbindelse med en efterfølgende uddannelse af respondenterne bør området tillægges opmærksomhed.

10.3 Samlet konklusion

Delkonklusionen på problemformuleringens første underpunkt er behandlet i afsnit 6.1.2. I kapitlet er det vist, at der kan opstilles et analyse- og uddannelsesmiljø med en SAP HANA Express database. Miljøet dækker behovet i forbindelse med analyserne inden for dette projekt, og det vil efterfølgende kunne anvendes i forbindelse med kommende uddannelsesaktiviteter. Problemformuleringens første underpunkt er således besvaret, og forudsætningen for besvarelse af de to øvrige underpunkter er indfriet.

Delkonklusionen på problemformuleringens andet underpunkt er behandlet i afsnit 7.5. I kapitlet er der gennemført en undersøgelse af sproget SAP HANA SQLScript med spørgeskemaet fra [9]. I undersøgelsen er der lagt vægt på forekomne fejl og områder, der var vanskelige at forstå. Begge dele med henblik på efterfølgende at kunne sammenholde resultaterne med dem, der kan opnås med Discount Method for Programming Language Evaluation, der indgår i problemformuleringens tredje underpunkt. Andet underpunkt er altså besvaret og formålet indfriet.

I forhold til problemformuleringens tredje underpunkt lægges det til grund, at "Cognitive Dimensions" er en valid metode til analyse af et programmeringssprog og dermed også kan anvendes til at afdække de problematiske områder, som kræver særligt indsats i forbindelse med uddannelse af programmører.

Projektet har vist, at tilsvarende resultater kan opnås ved brug af "Discount Method for Programming Language Evaluation".

Discount Metoden har vist sin styrke i forhold til at afdække manglende kendskab til og hermed aflede uddannelsesbehovet inden for:

- Kosmetiske problemer, herunder fx brug af tegnsætning og definition af typer.

- Seriøse problemer, herunder problemer med den logiske implementering af kode.

Discount Metoden har ikke i samme omfang kunne anvendes i forhold til at belyse uddannelsesbehov i relation til de mere komplekse områder, hvor der fordres en ny måde at tænke på, eller hvor udviklingsværktøjet (HANA Studio i den konkrete case) i høj grad støtter med udarbejdelse af kode fx i form af templates. Det tillægges ikke mangler ved Discount Metoden men det faktum, at Cognitive Dimensions blev anvendt til at analysere SAP HANA SQL Script efter ca. 50 timers indgående arbejde med sproget - ikke 60 minutter, som det var tilfældet under forsøgene med de 6 respondenter.

Problemformuleringens tredje underpunkt ses således også at være indfriet. I forhold til problemformuleringens tre underpunkter konkluders det samlet inden for rammerne af den konkrete case at:

Discount Method for Programming Language Evaluation kan bruges til at analysere og fastlægge uddannelsesbehovet ved overgangen fra ét programmeringssprog til ét andet.

I forhold til projektets case vurderes det, at der er opnået et særdeles godt indblik i uddannelsesbehovet hos de programmører i Forsvaret, der forventes at skulle videreuddannes og kunne arbejde med SAP HANA SQLScript for at kunne udnytte de teknologiske muligheder, der ligger i HANA teknologien inden for Business Intelligence.

Discount metoden har særligt bidraget til at kunne differentiere programmørerne i grupper med forskellige uddannelsesbehov. Det vil medvirke til, at der kan iværksættes uddannelsestiltag, som vil være tilpasset den enkelte.

Respondenternes og facilitatorens samlede anvendte "konfrontationstid" var knap 25 arbejdstimer. Hvis facilitator i forvejen er godt bekendt med detaljerne i det nye sprog, og der i forvejen findes et Cheat Sheet eller lignende til sproget, så kræver brugen af Discount Metoden derudover "blot", at der udarbejdes et opgaveark og den fornødne tid til den efterfølgende analyse af respondenternes opgavebesvarelser.

Med de givne forudsætninger, er der tale om en relativ beskeden indsats i forhold til resultatet. Metoden lever med andre ord op til discount tilgangen – også når den bruges til at vurdere uddannelsesbehov ved omstilling af programmører fra ét programmeringssprog til ét andet.

11 Usikkerhed

I dette kapitel behandles mulige usikkerheder i forbindelse med forsøget. Der fremhæves nogle oplagte kilder til usikkerhed fra før, under og efter det egentlige forsøg, uden at kapitlet nødvendigvis er udtømmende.

11.1 Usikkerhed før forsøg

Forfatteren havde ingen forudgående erfaring med SAP HANA SQLScript. Det kan have givet anledning til følgende usikkerheder.

Forfatteren brugte ca. 50 timer på at tilegne sig sproget. Det vurderes at være nok i forhold til sprogets basale elementer, men det er usikkert, om der kan være udeladt mere komplekse men relevante emner.

I forlængelse heraf kan analysen med Cognitive Dimensions mangle områder, der burde have være afdækket.

Til gengæld må det nok slås fast, at hverken Cheat Sheet eller Task Sheet dækker alle facetter af SAP HANA SQLScript. Det ligger i sagens natur for de to produkter, at de ikke skal dække sprogets fulde omfang.

Usikkerhederne skal imidlertid holdes op imod formålet; nemlig at gennemføre forsøg af en times varighed med respondenter, der stort set ikke har forudgående kendskab til sproget. Set i det lys har de nævnte usikkerheder næppe haft indflydelse på resultatet.

11.2 Usikkerhed under forsøg

Da forfatteren samtidigt var facilitator under de praktiske forsøg, kan det også have givet anledning til usikkerheder.

Facilitatoren havde selv kort tid forinden forsøgene selv gennemført analyse af sproget ved hjælp af Cognitive Dimensions. Facilitatoren kan således have opfattet respondenternes signaler under forsøget ud fra egne nylig gjorte erfaringer.

Selvom facilitator undervejs i forsøgene tilstræbte at spørge objektivt, kan det ikke udelukkes, at dialogen under forsøgene har været farvet af facilitators egen erfaring.

Endeligt er der arbejdsmæssig relation mellem respondenterne og facilitator, hvor facilitator kan betragtes som arbejdsgiver for respondenterne. Der er derfor en mulighed for, at respondenterne bevidst eller ubevidst har svaret på spørgsmål ud fra en formodning om, hvad facilitator ønskede at høre.

Undersøgelse af de nævnte usikkerheder kunne nok udgøre et større selvstændigt arbejde, der falder uden for rammerne af dette projekt. Det vurderes, at usikkerhederne ikke har haft indflydelse på konklusionerne, men vurderingen beror alene på et subjektivt skøn.

11.3 usikkerhed efter forsøg

På samme måde som under forsøget kan forfatterens efterfølgende analyse have været præget af egne erfaringer. Det må i den forbindelse konstateres, at projektet lå stille i en længere periode, og at forfatteres egne erfaringer derfor ikke længere var i frisk erindring, da analysen blev gennemført.

Under analysen viste det sig, at optagelser af skærmen var fuldt anvendelige, men at det i enkelte tilfælde ikke var muligt at tyde, hvad respondenten sagde. Respondenter fulgte i høj grad opfordringen til at tænke højt, og det var typisk i den sammenhæng, at der forekom mumlen eller usammenhængende sætninger, som ikke kunne tydes. Når disse situationer blev sammenholdt med skærmaktiviteterne, var respondenternes intention dog oftest forståelig, og det vurderes ikke, at de få korte udfald har haft indflydelse på resultatet.

12

Perspektivering og praktik

12.1 Perspektivering i forhold til det konkrete forsøg

Projektet har vist, at Discount Method of Language Evaluation ikke er begrænset til anvendelse i forbindelse med evaluering af nye computersprog under deres tilblivelse, men at den også kan anvendes til at fastlægge uddannelsesbehovet ved programmører, der skal tage et nyt computersprog i brug.

Forsvaret er i skrivende stund i gang med et større migreringsprojekt for DeMars Business Intelligence systemet. Som beskrevet i afsnit 4.3.3 er den underliggende database allerede opgraderet til HANA teknologi. I forbindelse med at data flyttes fra en traditionel diskbaseret database til en in-memory database, er der også behov for at omlægge data til en ny datamodel. Det er netop det, der nu gennemføres i et selvstændigt migreringsprojekt. Projektet forventes afsluttet i oktober 2020, hvorefter programmørernes næste større samlede aktivitet er at påbegynde anvendelse af SAP HANA SQLScript. I den forbindelse er det oplagt at anvende erfaringen fra projektet. Siden undersøgelserne i projektet blev gennemført, er der opstået nogle forhold, som bør inddrages i det videre arbejde.

- Teamet af programmører er blevet udvidet med yderligere en person. Det er oplagt at lade denne arbejde med Task Sheet på samme måde som de øvrige for at skabe et opdateret overblik over det samlede uddannelsesbehov.

- En af programmørerne fra det oprindelige forsøg har i mellemtiden arbejdet på at blive certificeret inden for SAP HANA i relation til Business Warehouse[1]. Erfaringerne i den forbindelse bør inddrages. Det kunne fx ske ved at lade programmøren gense sin egen optagelse af skærmaktivitet og lyd fra forsøget, og lade ham vurdere, hvordan resultaterne passer med erfaringerne fra uddannelsen som forberedelse til certificeringen.

[1]SAP HANA SQLScript udgør en delmængde af certificeringen.

12.2 Praktisk refleksion

Det er en grundlæggende præmis i Discount Metoden, at den ikke må være ressourcekrævende at gennemføre , tværtimod skal det være en billig metode. I den forbindelse er der gjort nogle erfaringer, som kan indgå i overvejelserne for den fremtidige anvendelse af metoden. Erfaringerne kan forekomme banale, men alligevel kan de være med til at smidiggøre kommende brug af metoden og reducere indsatsen frem mod resultaterne.

- I projektet er det valgt at transskribere forsøgene og interviews med respondenterne. Metoden forskriver ikke, at interviews skal transskriberes. Valget er sket, idet undersøgelsen indgår i et masterprojekt. Transskribering sikrer en minutiøs gennemgang af interviews, men det er en arbejdstung proces. En eller flere gennemgange af det indsamlede materiale fra hvert interview er helt klart en nødvendighed. Spørgsmålet er, om ikke det er nok at notere delkonklusionerne undervejs i gennemgangen af optagelserne i stedet for at transskribere hele forsøget.

- De første 10-15 minutter af hvert interview bruges på, at facilitatoren fortæller, hvad der skal ske. Det er med til at skabe en god stemning, men det giver ikke megen værdi at transskribere det efterfølgende. Det kunne også overvejes at skrive det ned, og fx sende det til respondenten dagen før interviewet. Det kunne sikre ens budskab og lette den efterfølgende behandling. Alternativt kunne indledningen indspilles som en kort video, som respondenten og facilitatoren kunne gennemse i fællesskab.

- Endeligt var der nogle meget lavpraktiske erfaringer:
  1) Brug en ekstern mikrofon. I projektets forsøg blev den indbyggede mikrofon i en bærbar computer anvendt, og samtidigt blev der anvendt en mobiltelefon som backup. Lydkvaliteten var svingende, og det besværliggjorde de efterfølgende analyser.
  2) Skaler brugerens billede på computeren, så det fylder hele skærmen. Når det efterfølgende skal analyseres, kan det vises i et mindre vindue og samtidigt være læseligt. På den måde kan der arbejdes i flere vinduer samtidigt i den efterfølgende analyse.

Kilder

[1] Svetomir Kurtev, Tommy Aagaard Christensen og Bent Thomsen. „Discount Method for Programming Language Evaluation“. I: *Plateau' 16, Amsterdam, Netherlands — November 01* (2016).

[2] Andreas Stefik og Stefan Hanenberg. „Methodological Irregularities in Programming-Language Research“. I: *Computer, Volume 50, Issue 8* (2017).

[3] Studienævn for Datalogi og Studienævn for Elektronik og IT Aalborg Universitet Det Naturvidenskabelige Fakultet. *Studieordning for Masteruddannelsen i IT, Linjen i Softwarekonstruktion*. 2016.

[4] Finansministeriet. *Finansloven 2019*. 2018. URL: `https://www.fm.dk/publikationer/2019/finansloven-for-2019`.

[5] Bill Chambers og Matey Zaharia. *Spark The Definitive Guide*. 2018.

[6] SAP. *Code push-down in ABAP Development*. 2018. URL: `http://www.beginners-sap.com/code-push-abap-development/` (bes. 04.05.2019).

[7] SAP. *SAP HANA SQLScript Reference*. 2018.

[8] DatabaseJournal.com Staff. *Procedural Versus Declarative Languages*. Database Management & Programming News, Articles & Tutorials for Database Administrator. URL: `https://www.databasejournal.com/sqletc/article.php/1408491/Beginning-SQL-Programming-Pt-2.htm` (bes. 13.05.2019).

[9] Alan F. Blackwell og Thomas R.G. Green. „A Cognitive Dimensions Questionary“. I: *alan.blackwell@cl.com.ac.uk & greenery@ntlworld.com* (2007).

[10] Alan F. Blackwell og Thomas R.G. Green. „A Cognitive Dimensions Questionary Optimized for Users“. I: *12th Workshop of the Psychology of Programming Interest Group, Cozenza Italy, April 2000, www.ppig.org* (2000).

[11] Thomas Green og Alan Blackwell. *Cognitive Dimensions of Information Artefacts: a tutorial*. 1998.

[12] Steven Clarke Microsoft Corporation. „Evaluating a new programming language“. I: *G.Kadoda (ed) Proc. PPIG 13* (2001).

[13] SAP e-book (kursusmateriale til SAP kursus HA150). *SAP HANA 2.0 SPS03 - SQL and SQLScript for SAP HANA*. 2018.

[14] Udemy. *Udemy online kursus: Time4HANA (T4H) - SAP HANA SQL & SQL Scripting*. 2019. URL: `https://www.udemy.com/user/time4hana/` (bes. 05.06.2019).

[15] Udemy. *Udemy online kursus: SAP HANA SQL Scripting - Step 2*. 2019. URL: https://www.udemy.com/course/sap-hana-sql-scripting-step-2/ (bes. 08.08.2019).

Bilag

A

Task Sheet

Denne tabel indgår i nogle af opgaverne nedenfor:

Tabel A.1: Personel.

Navn VARCHAR(30)	**Kategori** VARCHAR(30)	**Titel** VARCHAR(30)	**Timeloen** INT
Ole Olsen	Militær	Kaptajn	290
Hanne Hansen	Civil	Fuldmægtig	310
Jens Jensen	Militær	Oversergent	240
Niels Nielsen	Militær	Kaptajn	312
Jørgen Jørgensen	Civil	Bogholder	300
Peter Petersen		Præst	305

Tabellen er placeret i det aktive Schema, der har navnet "FORSVARET". I Schemaet findes også "FORSVARET_PAC", der er et repository, hvor "Design-time" kode kan placeres.

A.1 Opgave 1

Stamværdi på miltært materiel omfatter ofte vægt. Nogle gange i kg andre gange i pund (lbs). Der er ofte behov for at omregne mellem enhederne.

Skriv en rutine, der har én input parameter og én outputparameter. Som input modtager rutinen en vægt i pund (lbs) som output skal rutinen levere vægten i kg.

$$kg = lbs/2.2046$$

fx:

$$7,3lbs = 3,3113kg$$

A.2 Opgave 2

Personellet i tabel A.1 har ind imellem overarbejde. Da personellet har forskellige overenskomster, udbetales overarbejde efter forskellige regelsæt.

Skriv en rutine, der har to input parametre "Kategori" og "Timeloen" fra tabel A.1. Rutinen skal have et output.

Hvis kategorien er "Militær" skal rutinens output være Timeløn ganget med 1,4. Hvis kategorien er er "Civil" skal rutinens output være Timeløn ganget med 1,6. Hvis kategorien hverken er "Militær" eller "Civil" skal output være timelønnen uændret.

A.3 Opgave 3

Skriv en rutine, der beregner $x!$ (x fakultet).

Fx

$$5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$$

Rutinen har en input parameter – nemlig x. Rutinen skal have et output $x!$. Bemærk, at $x!$ kan blive et meget stor helt tal.

A.4 Opgave 4

I forbindelse med lokallønsforhandlinger er der ofte behov for at sammenligne ansatte med samme samme "Titel". Se på tabel A.1. Skriv en rutine der har én inputparameter – Titel. Rutinen skal returnere en tabel over personer med titlen som output. Outputtabellen skal indeholde kolonnerne Navn, Kategori og Timeløn.

Tabel A.2: Personel med titlen kaptajn.

Navn	**Kategori**	**Timeloen**
Ole Olsen	Militær	290
Niels Nielsen	Militær	312

A.5 Opgave 5

Skriv en rutine, der tilføjer et antal tilfældige heltal i en tabel med navnet: "RandomTable" - et tilfældigt tal i hver række.

Tabellen "RandomTable" findes i forvejen, tabellen har én kolonne: "RandomNumnber".

Rutinen skal have tre inputparametre:

- Antal rækker med tilfældige tal, der skal tilføjes
- Nedre grænse for de tilfældige tal (helt tal)
- Øvre grænse for de tilfældige tal (helt tal)

Hint: RAND giver et tilfældigt tal.

$$0 \leq RAND() < 1$$

Når rutinen er kørt med parametrene (6, 2, 5), kan RandomTable fx se sådan ud:

Tabel A.3: RandomTable

	RandomNumber
1	2
2	3
3	2
4	4
5	4
6	3

B

Eksempelvis løsning

B.1 Eksempelvise løsninger på Task Sheet

Nedenfor findes forslag til eksempelvis løsninger på opgaverne i Task Sheet. I forhold til løsninger bemærkes følgende:

- Den enkelte løsning er blot en ud af mange mulige løsninger på opgaven.
- Løsningerne udgøres af funktionsduelig kode.
- Løsningerne er testet i SAP HANA Studio imod en SAP HANA Express 2.0 database.
- Løsningerne indholder i begrænset omfang supplerende kode, der sætter systemet op til at opgavekoden efterfølgende kan eksekveres. Tilsvarende er der i nogle tilfælde ekstra kode, der kan bruges til kalde den faktiske løsning.
- Løsningerne er minimalistiske. Der foretages således fx ikke kontrol af inputparametre, fejlhåndtering eller lignende.
- Løsningerne følger Forsvarets retningslinjer for programmering. Det betyder blandt andet, at variabelnavne m.m. er på engelsk, mens kommentarer kan være på både dansk og engelsk.
- Oprettelse af tabellen "Personel", der præsenteres som det første i Task Sheet, kan ses i løsningen for Opgave 4.

B.1.1 Løsning – Opgave 1

```
1 -- Opgaven løses med brug af en User Defined Function -
     Scalar Function
2
3 DROP FUNCTION lbsToKg;
4
5 -- Selve opgave 1
```

```
6  CREATE FUNCTION lbsToKg(lbs DECIMAL(10,2))
7  RETURNS result DECIMAL(10, 4) AS
8  BEGIN
9    result := :lbs / 2.2046;
10 END;
11
12 -- Kald af funktionen
13 SELECT lbsToKg(7.3) FROM DUMMY;
```

Listing B.1: Eksempelvis løsning af Opgave 1.

B.1.2 Løsning – Opgave 2

```
1  -- Opgaven løses med brug af en User Defined Function -
       Scalar Function
2  -- Opgaven gør brug af Conditionals - Cheat punkt 3
3
4  DROP FUNCTION overtime;
5
6  -- Selve opgave 2
7  CREATE FUNCTION overtime(cat VARCHAR(15), salery INT)
8  RETURNS overtimeSalery DECIMAL(10,2) AS
9  BEGIN
10   IF :cat = 'Militær' THEN
11     overtimeSalery := :salery * 1.4;
12   ELSEIF :cat = 'Civil' THEN
13     overtimeSalery := :salery * 1.6;
14   ELSE
15     overtimeSalery := :salery;
16   END IF;
17 END;
18
19 -- Test af funktionen
20 SELECT overtime('Militær', 100) FROM DUMMY;
21 SELECT overtime('Civil', 100) FROM DUMMY;
22 SELECT overtime('Værnepligtig', 100) FROM DUMMY;
```

Listing B.2: Eksempelvis løsning af Opgave 2.

B.1.3 Løsning – Opgave 3

```
1  -- Opgaven løses med brug af en User Defined Function -
       Scalar Function
2  -- Opgaven gør brug af Loops - Cheat punkt 3
3
4  DROP FUNCTION fakultet;
5
```

```
6  -- Selve opgave 3
7  CREATE FUNCTION fakultet(x INT)
8  RETURNS result BIGINT AS
9  BEGIN
10   DECLARE i INT = 1;
11   result := 1;
12   FOR i IN 1 .. :x DO
13     result := :result * :i;
14   END FOR;
15 END;
16
17 -- Test af funktionen
18 SELECT fakultet(5) FROM DUMMY;
```

Listing B.3: Eksempelvis løsning af Opgave 3.

B.1.4 Løsning – Opgave 4

```
1  -- Opgaven løses som en User Defines Table Function
2  -- UDTF skabes "design time" og skal derfor placeres i
      et Repository i et Schema
3  -- Her er Schema = "FORSVARET" og Repository =
      "FORSVARET_PAC"
4
5  CREATE SCHEMA FORSVARET
6  SET SCHEMA FORSVARET
7  GRANT SELECT ON SCHEMA FORSVARET TO _SYS_REPO WITH GRANT
      OPTION
8
9  DROP TABLE "Personel";
10 CREATE TABLE "Personel"(Navn VARCHAR(30), Kategori
      VARCHAR(30), Titel VARCHAR(30), Timeloen INT);
11
12 TRUNCATE TABLE "Personel";
13 INSERT INTO "Personel"(Navn, Kategori, Titel, Timeloen)
      VALUES('Ole Olsen', 'Militær', 'Kaptajn', 290);
14 INSERT INTO "Personel"(Navn, Kategori, Titel, Timeloen)
      VALUES('Hanne Hansen', 'Civil', 'Fuldmægtig', 310);
15 INSERT INTO "Personel"(Navn, Kategori, Titel, Timeloen)
      VALUES('Jens Jensen', 'Militær', 'Oversergent', 240);
16 INSERT INTO "Personel"(Navn, Kategori, Titel, Timeloen)
      VALUES('Niels Nielsen', 'Militær', 'Kaptajn', 312);
17 INSERT INTO "Personel"(Navn, Kategori, Titel, Timeloen)
      VALUES('Jørgen Jørgensen', 'Civil', 'Bogholder', 300);
18 INSERT INTO "Personel"(Navn, Kategori, Titel, Timeloen)
      VALUES('Peter Petersen', 'Militær', 'Konstabel', 240);
19
```

```
20 -- Selve opgave 4
21 -- Koden kan ikke oprette som et SQL kald, men skal
      udvikles i et Repository
22 FUNCTION "FORSVARET"."FORSVARET_PAC::lokalLoen" (funk
      VARCHAR(30))
23 RETURNS TABLE (Navn VARCHAR(30), Kategori VARCHAR(30),
      Timeloen INT)
24 LANGUAGE SQLSCRIPT
25 SQL SECURITY INVOKER AS
26 BEGIN
27   RETURN
28   SELECT Navn, Kategori, Timeloen
29   FROM "FORSVARET"."Personel"
30   WHERE Titel = :funk;
31 END;
32
33 -- UDTF kan derimod kaldes fra SQL sådan:
34 SELECT * FROM
      "FORSVARET"."FORSVARET_PAC::lokalLoen"('Kaptajn');
```

Listing B.4: Eksempelvis løsning af Opgave 4.

B.1.5 Løsning – Opgave 5

```
1  -- Opgaven løses med brug af en Stored PROCEDURE
2  -- Cheat Sheet punkt 7
3
4  -- Klargør tabel
5  DROP TABLE RandomTable;
6  CREATE TABLE RandomTable(RandomNumber INT);
7
8  -- Det efterfølgende kan køres som en blok
9
10 -- Slet alle data i tabellen, men ikke tabellen
11 TRUNCATE TABLE RandomTable;
12
13 -- Slet procedure
14 DROP PROCEDURE insertRows;
15
16 -- Opret procedure
17 CREATE PROCEDURE insertRows(number INT, lower INT, upper
      INT)
18 LANGUAGE SQLSCRIPT
19 SQL SECURITY INVOKER
20 AS
21 BEGIN
22   DECLARE i INT;
```

```
23    FOR i IN 1 .. :number DO
24      INSERT INTO RandomTable(RandomNumber) VALUES(lower +
          RAND() * (upper - lower));
25    END FOR;
26  END;
27
28  -- Eksekver procedure
29  CALL insertRows(6, 2, 5);
```

Listing B.5: Eksempelvis løsning af Opgave 5.

C

Cheat Sheet

C.1 Udarbejdelse af Cheat Sheet

Det har ikke været muligt at finde eksisterende Cheat Sheet for SAP HANA SQLScript. Det har derfor været nødvendigt at udarbejde et Cheat Sheets til forsøgene.

Cheat Sheet er en særdeles komprimeret udgave af de væsentligste informationer fra:

- SAP HANA SQLScript Reference, 2018.
- SAP HANA 2.0 SPS03 - SQL and SQLScript for SAP HANA, SAP e-book (kursusmateriale til SAP kursus HA150), 2018.
- SAP HANA SQL & SQL Scripting, Udemy online kursus: Time4HANA (T4H), 2019, www.udemy.com.
- SAP HANA SQL Scripting - Step 2, Udemy online kursus, 2019, www.udemy.com.

Cheat Sheet indeholder en blanding af oversigter og små stykker eksekverbar HANA SQLScript. Sidstnævnte er afprøvet i SAP HANA Studio imod en SAP HANA Express 2.0 database.

SAP HANA SQLScript Cheat Sheet

1. Declarative programming.

Querying data from a table:

```
-- Query data in columns c1, c2 from table t
SELECT c1, c2 FROM t;

-- skeleton for complex query
SELECT AUTHOR AS AUTH, CATEGORY, COUNT(*)
FROM "Books"
WHERE CATEGORY = 'Printed'
GROUP BY CATEGORY, AUTHOR
HAVING COUNT(*) >= 2
ORDER BY AUTHOR;
```

Aggregate functions:

```
AVG, COUNT, SUM, MAX, MIN
```

Querying data from multiple tables:

```
SELECT C1, C2
FROM t1
LEFT JOIN t2 ON t1.c1 = t2.c4;
```

Managing tables:

```
-- Create a table with 3 columns
CREATE TABLE t(id INT PRIMARY KEY,
               name VARCHAR(10) NOT NULL,
               price INT DEFAULT 0);

-- Add a new column to the table
ALTER TABLE t ADD (newcolumn INT);

-- Delete table from database
DROP TABLE t;
```

Modifying data:

```
-- Insert a new row with data
INSERT INTO t(id, name, price)
VALUES (5, 'John Doe', 10);

-- Modify a row
UPDATE t
SET name = 'Freddy',
    Price = 20
WHERE id = 5;

-- Delete some rows
DELETE FROM t
WHERE id >= 4;
```

2. Data types.

Scalar Data Types:

Numeric Types	TINYINT SMALLINT INTEGER BIGINT DECIMAL(t,f)	8-bit unsigned 16-bit signed 32-bit signed 64-bit signed Fixed point dec. (total, frac)
Numeric String Types	VARCHAR(n) NVARCHAR(n) ALPHANUM(n)	n max # of char unicode
Data-Time Types	TIMESTAMP SECONDDATE DATE TIME	
Binary Types	VARBINARY(n)	
Large Object Types	CLOB NCLOB BLOB	Ascii Unicode Binary
Spatial Types	ST_GEOMETRY	
Boolean Types	BOOLEAN	TRUE/FALSE

Can optionally be initialized with their declaration:

```
DECLARE local_var INT;
DECLARE c INT = 5;
DECLARE a INT DEFAULT 0;
```

Assignment done using "=":

```
local_var = :c + :a + 3;
```

Variables can be referenced by using ":" as prefix in the variable.

Table Types:

These types are used by the user to define parameters for procedures representing tabular results.

```
CREATE TYPE myType AS
       TABLE (id INTEGER, name VARCHAR(10));
```

Can optionally be initialized with their declaration:

```
DECLARE temp TABLE(n int);
DECLARE outTab MY_TABLE_TYPE;
DECLARE inTab MY_TABLE_TYPE =
              UNNEST (:arr) AS (i);
DECLARE temp CONSTANT MY_TABLE_TYPE
              DEFAULT SELECT * FROM TABLE;
```

Assignment done using "=":

```
outTab = SELECT * FROM :inTab;
```

3. Imperative programming.

Conditionals:

```
IF :in = 2 THEN
   out := "Exactly Two";
ELSEIF :in > 2 THEN
   out := "Greater than two";
ELSE
   out := "Less than two";
ENDIF;
```

Loops:

```
-- Loop a specific number of times
FOR x IN 0 .. 10 DO
   IF :x < 3 THEN
      CONTINUE; -- Start next iteration
   END IF;

   IF :x = 5 THEN
      BREAK; -- Stop processing and exit loop
   END IF;
END FOR;

-- Loop as long as…
WHILE :i < 10 DO
   i := :i + 1;
END WHILE;
```

SAP HANA SQLScript Cheat Sheet

4. User Defined Functions (scalar functions).

Limitations:

- UDF can only contain imperative programming.
- UDF can only return a single Numeric Type.

Examples:

```
-- UDF with one inputparameter
-- Convert hours to minutes
CREATE FUNCTION Convert_Hours(im_hours INTEGER)
RETURNS ex_result DEC(5, 2) AS
BEGIN
  ex_result := :im_hours * 60;
END;


-- UDF with two input parameters
-- Convert hours to minutes or days
CREATE FUNCTION Convert_Hours2(im_hours INT,
im_to VARCHAR(1))
RETURNS ex_result DEC(5, 2) AS
BEGIN
  IF :im_to = 'm' THEN
    ex_result := :im_hours * 60;
  ELSEIF :im_to = 'd' THEN
    ex_result := :im_hours / 24;
  ELSE
    ex_result := :im_hours;
  END IF;
END;
```

5. Useful constants and built-in functions.

```
SELECT 'Hello World!' FROM DUMMY;
SELECT CURRENT_DATE FROM DUMMY;
SELECT CURRENT_USER FROM DUMMY;
SELECT CURRENT_TIME FROM DUMMY;
SELECT 2 + 7 FROM DUMMY;
SELECT NOW() FROM DUMMY; -- DateTime in millis.
SELECT UPPER('to upper case') FROM DUMMY;
SELECT RAND() FROM DUMMY; -- Random 0 to 1.
SELECT SUBSTR('12xy5', 3, 2) FROM DUMMY; -- xy
SELECT NEXT_DAY(NOW()) FROM DUMMY;
```

DUMMY is used for demonstration. See section 8 for use of DUMMY.

6. User Defined Table Functions.

Defining User Defined Table Function:

```
FUNCTION "mySchema"."myPac::tfBooks"
                    (cat VARCHAR(10))
RETURNS TABLE (bookId VARCHAR(3),
               category VARCHAR(10),
               price DECIMAL(10, 2))
LANGUAGE SQLSCRIPT
SQL SECURITY INVOKER AS
BEGIN
  RETURN
  SELECT bookId, category, price
  FROM "mySchema"."Books"
  WHERE category = :cat;
END;
```

User Defined Table functions are created design time, hence they must reside inside a repository (Package) inside a Schema, therefore the use of “myShcema” and “myPac”. Noe: tfBooks is the name of the function.

Use of Table Types:
If a Table Type is create inside mySchema by:

```
CREATE TYPE bookType AS
            TABLE(bookId VARCHAR(3),
            category VARCHAR(10),
            price DECIMAL(10, 2));
```

The function would look like this:

```
FUNCTION "mySchema"."myPac::tfBooks"
                    (cat VARCHAR(10))
RETURNS bookType
LANGUAGE SQLSCRIPT
SQL SECURITY INVOKER AS
BEGIN
  RETURN
  SELECT bookId, category, price
  FROM "mySchema"."Books"
  WHERE category = :cat;
END;
```

Calling the function:
Books of the category ‘pdf’ in the Books table can be found by:

```
SELECT * FROM
"mySchema"."myPac::tfBooks"('pdf');
```

7. Stored Procedures.

Examples:

```
-- Stored procedure with one input parameter and
-- change an existing table as the outcome.
CREATE PROCEDURE filterBooks(
       IN cat VARCHAR(1),
       OUT result TABLE(bookId VARCHAR(3),
                        category VARCHAR(10),
                        price DECIMAL(10, 2)))
LANGUAGE SQLSCRIPT
SQL SECURITY INVOKER
READS SQL DATA AS
BEGIN
  DECLARE x VARCHAR(10);
  IF :cat = 'p' THEN
    x := 'pdf';
  ELSEIF :cat = 'e' THEN
    x := 'e-book';
  ELSE
    x := 'Printed';
  END IF;

  RESULT = SELECT bookId, category, price
  FROM "books" WHERE CATEGORY = :x;
END;
```

Calling the procedure:

```
CALL filterBooks('p', ?);
```

Note that a procedure can be used the same way as a Table Function, but that the Table Function is made design time, hence it can be transported in a Dev. → Q/A → Prod. setup.

8. Miscellaneous.

Use of DUMMY:
DUMMY is a build-in table in HANA, that contains exactly one column and one row.

```
-- See scalarfunction Convert_Hours() above
Select Convert_Hours(3) FROM DUMMY;

-- See scalarfunction Convert_Hours2() above
Select Convert_Hours2(48, 'd') FROM DUMMY;
```

See also section 5 for use of DUMMY.

D

Transskriberinger

D.1 Transskriberinger

På de efterfølgende sider findes transskribering af filerne:

- Respondent nummer 1: 2019-09-18_10-27-00.mp4
- Respondent nummer 2: 2019-09-18_14-12-03.mp4
- Respondent nummer 3: 2019-09-24_10-34-33.mp4
- Respondent nummer 4: 2019-10-01_11-33-48.mp4
- Respondent nummer 5: 2019-10-03_10-40-58.mp4
- Respondent nummer 6: 2019-10-04_11-11-35.mp4

Transskribering af forsøg med IBM medarbejder nummer 1.

Fil: 2019-09-18_10-27-00.mp4

Gennemført 2019-09-18.

Tid	Tale	Resultat og bemærkninger
0:43 Indledning	F: Nu skal du se, den her, den så du sidst. I: Ja. F: Det du bliver udsat for nu, tager en god times tid, og du får et cheat sheet lige om lidt. Og det får du lidt tid til at læse. Jeg tror at "B" brugte en 4-5 min. Du skal ikke gå op i alle detaljerne og tro, at du skal lære det hele udenad, men bare orientere dig om, hvor du cirka kan finde hvad. Og så får du et opgavesæt, og så er det bare at klø på stille og roligt. Udviklingsværktøjet er noget af det mest avancerede, der findes – Wordpad. Jeg har sat det op, så det gemmer på Q-drevet. Snaglt kører også i baggrunden. I: Den blinker i hvert fald. F: Der er nok flere opgaver end du kan nå igennem, i hvert fald hvor det bliver til 100% eksekverbar kode. Det ville også være det værste, der kunne ske, at du nåede det hele, og at det kunne køre uden fejl, uden at der har været usikkerheder undervejs, for det er også dem, jeg kigger efter. I: Men du ved jo ikke, om det kan køre. F: Jeg kunne jo være så fræk at gå hjem og afprøve det. Jeg kan godt spotte fejl og usikkerheder, jeg har trods alt brugt en hel del timer på det. Jeg har været sproget igennem efter en anden analysemetode, og nu jeg gerne se, om Discount Metoden kan afdække nogle af de samme ting. Så det skulle gerne være sådan, at noget af det der irriterede mig, eller noget af det jeg glemte, eller noget af det jeg syntes var nemt… at jeg kan spotte noget af det samme, i det du gør – velvidende at der er forskellige forudsætninger. F: Du må meget gerne tænke højt undervejs. Du får som sagt lov til at læse cheat sheet i fred og ro. Derefter sætter jeg mig her (ved siden af), og så kan vi tale om det. Sig højt hvad du tænker, hvor er der noget i cheat sheet, du kan bruge,	I er forkortelse for medarbejderen fra IBM. F er forkortelse for Facilitator, dvs. den person, der forestår forsøget og interviewer respondenten. I Deltog i det samlede kick off møde, og virker indforstået med opgaven. Samtalen foregår i en hyggelig tone. Det er tydeligt at I og M har mødt hinanden tidligere. Korte bemærkninger som ja, hmm, grin m.m. er ikke udskrevet. I har gennemført HANA150 (et kursus i SAP HANA SQL Script).

Tid	Tale	Resultat og bemærkninger
	hvad driller, der kunne også være flere måde at løse en opgave på, det tager vi så en snak om, når vi er færdige. Der spørger jeg også kort til, om det ligner noget, du tidligere har arbejdet med samt dine erfaringer. I: Det ligner HANA150. F: Lige præcis – værsgo. F: Begynder du at have overblik over cheat sheet? I: Ja. F: Du har måske gravet dig ned (i detaljer). I: Ja, jeg har gravet mig ned. F: Prøv at se opgavearket her. Først er der en tabel, som bliver brugt i nogle af de senere opgaver. Opgave 1 er en opvarmningsopgave, hvor du stort set kan finde skellettet derovre (i cheat sheet). 2'eren, der kommer du skele flere steder i cheat sheet. Herefter er der opgaver inden for forskellige områder, men med stigende sværhedsgrad. Værsgo.	
9:27 **Opgave 1**	I: Opgave 1 er en konverteringsrutine, hvor jeg bare skal tage Input og gange med noget andet. Input er i pound, og det bliver så regnet om til vægt. Det vil sige, at jeg skal lave en funktion til det her, der bliver kaldt med det der. Jeg vil skyde på at jeg skal kigge i cheat sheet… og funktionerne de ligger heroppe. Skal jeg virkeligt skrive det hele? F: Ja, men det ser ud til, at du kan klare dig med fire linjer. I: (Skriver og tænker højt) Har vi et navn til den? Det har vi ikke. Lbs til kilo. Den skal have en input parameter. Hvis vi kaldet det lbs, og siger det skal være DECIMAL 10,2. Den kan justeres. Så skal vi have return og den returner vi ex_kg. Den kopierer vi til DECIMAL 10,2. Og så begin. Og så siger vi. Lig med lbs divideret med. Jeg håber, at det er i orden. Så retter vi lige input til in_. I: Hvis vi så lige tjekker op. Der skal være kolon foran variable. Der skal være kolon foran lig med.	Foretaget sikkert valg af DECIMAL som input og output. Definerer variable med in_ og out_ prefiks. Er måske i tvivl om brugen af forkortelserne DEC og INT for hhv. DECIMAL og INTEGER. Følger eksemplet i cheat sheet slavisk.

Tid	Tale	Resultat og bemærkninger
	Og semikolon til sidst efter END. Det burde være det!	
15:19 **Opgave 2**	I: (Læser, mumler og tænker højt). Okay (kopierer løsning fra opgave 1 og retter). I: Timelønskorrektion. Og den skal have kategorier – inputparameter kat. Så kategorina er VARCHAR – mon der skal være semikolon? Timeløn og den er INTEGER. Og så skal den returnere en der bliver ganget med 1,4 eller 1,6 – så burde der kunne være en decimal, lad os tage 10, 2. Hvis in_kat lig med civil – det går lidt stærkt – THEN Så sætter vi timeløn korrigeret – den må så blive lig med ind timeløn – gange – civil det er 1,4. ELSE gange ENDIF, hov det er i to ord. I: Og så vil jeg lige gøre sådan… Hvis in_kat er civil så er timeløn lig med timeløn gange 1,4. Hvis militær så gange 1,6. Og ellers så er det bare timeløn, så har vi også med hvis folk har noget… F: Ja, fx præsten. I: Ja, der hverken er det ene eller andet. I: Lad os se om det mangler det ene eller andet. Kan vi trykke på compile? (Smiler) F: Jeg tror nok det vil kunne kompilere, og de militære ville blive glade. I: Arh, jeg har byttet rundt. F: Det er en logisk fejl, det er ikke afgørende for undersøgelsen. I: Så, nu er de militære knap så glade.	Usikker på adskillelse af inputparametre – semikolon? Gennemlæser koden og foretager rettelser sikkert – støttet af cheat sheet.
23:00 **Opgave 3**	I: (kopierer opgave 1, og tænker herefter højt) Så laver vi en funktion – fakultet. Som vi kalder x, og det er en INTEGER. Den returner en anden INTEGER – fakultet INTEGER. Så skal vi lige loope igennem. Vi ser, om der er en loop (i cheat sheet). Så skriver vi WHILE kolon x større end nul, nej et DO.	I er sikker i gennemførelsen af opgaven, herunder implementering af logikken.

Tid	Tale	Resultat og bemærkninger
	Og så skal vi sige ex_fakultet lig med. Jeg glemte lige. Jeg sætter lige ex_fakultet til at være lig med et. Og så siger jeg, ex_fakultet skal være lig med ex_fakultet gange med in_x semikolon. Og så siger vi x kolon lig med kolon x minus et. Så løber den igennem. Så har vi først et ettal. Så siger vi fakultet det er lig med…. gange fem. Så har vi fem i ex_fakultet. Så sætter vi ned til fire. Så løber den igennem, så siger vi fem gange fire. Så har vi gange tre, gange to, en. Så exitter vi loopet. X skal hedde in_x. F: Jeg tror ikke, at det vil compile. I: Der mangler et semikolon der. F: Det ser fint ud nu. F: Kunne du løse det med andet end WHILE? I: Jeg kunne også have brugt en FOR loop, hvis jeg ville. Jeg kunne også have brugt variable anderledes… F: Enig. I: (læser opgave 3)	
27:56 **Opgave 4**	I: (tænker højt – utydeligt) En inputparameter, der hedder titel. Så skal den returnere en tabel, hvor siger den bare… (trommer i bordet). Normalt ville jeg jo bare have lavet sådan en almindelig… Jeg kan ikke huske – det kommer vist heromme (i cheat sheet), det vist bare en tablefunction… F: Lige nøjagtig. I: Function – og hvis det er myschema, så er det nok – forsvaret. Jeg vil mene, at jeg burde have lavet en create function myschema dot pac dot kolon kolon. F: Hvis den ikke findes i forvejen, så skulle man…	I arbejder koncentreret, og anvender i høj grad cheat sheet. Flere rettelser i koden indikerer, at I er usikker på brugen af schema og repository. I accepterer linjen ”SQL SECURITY INVOKER AS” som noget, der bare skal stå der.

Tid	Tale	Resultat og bemærkninger
	I: Så jeg kan skrive det inde i SQLen. F: Præcis. I: Function. (skriver og retter). Skal returne table. En avanceret tabel. (skriver og retter). Language SQLSCRIPT. SQL SECURITY INVOKER AS – det må bare være noget, der skal stå der. F: Man kan kalde de der scripts med forskellig rettigheder – enten brugerens eller programmørens. I: Ja okay, så vil man nok altid bruge invoker. Nogle gange er der noget, man blot vender sig til altid skal skrives. Metode. SELECT. Timeløn. FROM personel – med (kopierer). Og det kan nok diskuteres med dobbelt apostroffer, det er vist ikke altid nødvendigt. Det var titel lig med kolon. Det var min variabel in_titel. Så er spørgsmålet, om det vil compilere? Logikken er der i hvert fald. F: Der er et par fejl. Tabellen Personel ligger ude i skemaet og ikke i repository. Skemaet ligger i Forsvaret. I: Jaja, det er lige med at læse det rigtigt. Men igen er det sådan noget, hvor jeg ville blive slået over fingrene af compileren. F: Ja, du får igen hjælp her. I: Vi har det for godt til dagligt.	
38:15 **Opgave 5**	I: (Læser opgave 5 højt) Så den tabel findes i forvejen. Lige præcis denne der skal ikke returnere noget, så det er en procedure. Vi laver en ny – CREATE PROCEDURE, og den skal ha' input, det ser ud som jeg skal skrive in det fremgår ikke helt… nåh… Du kan vel godt have flere variable som input? F: Det kan du, og det fremgår her. Du angiver navnet og definitionen som her. I: Jeg er ikke helt sikker på, om jeg skal skrive IN foran dem alle sammen (input variable)?	I er ret sikker i valget af en procedure. I er usikker på beskrivelse af inputparametre til proceduren. I er usikker på scope for tabellen. I er usikker på tilføjelse af data til tabellen. I bruger noget tid på

Tid	Tale	Resultat og bemærkninger
	F: Det behøver du ikke. I: Hmm, så har jeg defineret mine ting ind, jeg har ikke noget output, fordi det er tabellen, og den skal stå der, den behøver ikke at blive defineret, den er defineret i forvejen. Jeg mener, at man skal huske, at hvis du bruger en variabel derinde, så skal den stå… eller også skal den være… nu laver jeg lige noget her, fordi jeg tror at, ja jeg tror at vi har… Det fremgår ikke helt klart, men normalt så plejer man skulle skrive, hvis du bruger en tabel… F: Prøv at se her. I: Ja, men det er ikke på den måde. Det er når vi laver AMPD procedurer så skal vi bruge USING. Nu står det ikke her, så jeg lader være, men igen det ville jeg få fortalt (af compileren) hvis det var. Og det bør du heller ikke skulle bruge i almindelig SQL. BEGIN. (trommer og mumler). Tabellen findes i forvejen og har en kolonne. Hvis jeg skal skrive en, der skriver et tilfældigt tal ned… Tilføjer, nåh OK. Så gør vi det bare lettere. F: Ja, det var et forsøg på at hjælpe. I: Der vil jeg nok sige. (skriver og retter koncentreret). Har vi en funktion, der hedder random? F: (Peger på hint i opgaveteksten). I: Giver den altid en mellem nul og et? F: Ja, den spytter altid noget ud, der kan være nul men aldrig et. Altså et decimaltal herimellem. I: Så du har ingen mulighed her for at sætte den… jeg vil godt have et helt tal dvs. at den kunne regne et helt tal ud for mig, men den leverer et decimaltal mellem nul og et. Det vil sige at, jeg kan tage decimaltallet, det er en procent, og så gange det, der står ikke hvor mange decimaler den er på. F: Den er på mange. I: Den er på mange, så skal jeg have heltalsværdien, det tal her? F: Hvis du tager det, der kommer ud og fx ganger med seks, så får du et tal mellem nul og fem komma ni, ni, ni…	logikken i proceduren. Det er næppe sprogafhængigt. I er usikker på hvordan en værdi castes fra DECIMAL til INTEGER, men er bevist om behovet.

Tid	Tale	Resultat og bemærkninger
	I: Jo, lige præcis dvs. procentsatsen gange, og skal jeg stadigt sørge for… F: Ja, du må nok gange og lægge til. I: Så det vil sige, at jeg kan (skriver). INSERT INTO randomtable hmm. Og herinde kan jeg så sige. RANDOM gange parentes upper minus minimum parentes slut plus lower_val). Det er så ikke helt svaret, for det giver stadigt et decimaltal. F: Det kommer an på. I: Jeg har et decimaltal, jeg ganger med, så ville jeg normalt have lavet en TOINT eller noget. F: Det kommer an på, hvordan tabellen er defineret. Du kan se at kolonnen bare er en integer. Så HANA laver det selv om. I: Så konverterer den selv. Så skal jeg. Jeg synes, at det er blevet lidt rodet. Normal så vil den bare løbe igennem ti gange, hvis ikke du skriver CONTINUE. (mumler gennemgang af koden). F: Jeg er ret sikker på, at det der ville virke. Hvad ville der ske, hvis du ikke satte default lig med et? I: Så ville den være nul. Så FOR starter med at sætte den til en, jah det vil være lige meget.	
52:10 (Afsluttende interview)	F: Du er igennem I. Du har tid i overskud. Det kommer an på hvor lang interviewer tager. Var der noget, der var nemt? I: Jeg vil sige det sådan, at det der cheat sheet vil jeg gerne have en kopi af. Fordi det skidegodt med sådant et opslagsværk. Nogle gange så glemmer man lige, og når det er et andet sprog. SQL Script er lidt anderledes, end hvis du bruger Oracle SQL. Og lige for at kunne få de sidste detaljer med og tjekke op. F: Det kan vi sagtens finde ud af. Der findes ikke et, så jeg har været nødt til at lave det selv. Det er bl.a. lavet ud fra det kursusmateriale du kender (SAP HANA 150) og et par andre SAP dokumenter. I: Det, jeg alternativt gør, er at google. Har man et (cheat sheet) er det stadigt hurtigere.	Kopi kan udleveres, når alle testspersoner har været igennem.

Tid	Tale	Resultat og bemærkninger
	F: Jeg legede også Calculation Views derhjemme, men der er man så afhængig af udviklingsværktøjet for at komme igennem alle de formkrav, der er, så det er ikke rigtigt til at sætte ind i et cheat sheet og opgaver som disse. I: Nej, men det basale er med som FOR løkke og WHILE løkke. De er meget gode lige at have med. F: Hvad tænker du om at springe mellem SQL og Imperativ med IF-THEN og FOR løkker, det jo to lidt forskellige verdener. I: Ja både og for vi gør jo det i dag i ABAP, at vi sidder og skriver lidt SQL inden i ABAP, det af så blot ABAP SQL. Det føles ikke helt fremmed, hvis vi ser generelt på det. Det var en tærskel, da jeg selv prøvede det første gang. Det var i forbindelse med TID (et stort projekt til styring af arbejdstid, overenskomster, frihed, ydelser, vagter m.m. i Forsvaret). Der anvendte jeg lignende. Når man kommer ind i det, så er det godt. Det er sejt, at du kan declarere en variabel som en tabel, ved blot at skrive lig med SELECT et eller andet, og så er den declareret…. *Drøftelse af hvordan cheat sheet kan udbygges.* F: Det er noget med, at du har 20 års programmeringserfaring eller mere. I: Jo, det må vi vel sige. Så vi må håbe, at jeg ikke falder fuldstændig igennem. F: Jeg har en forventning om, at du måske er et af yderpunkterne. I: Jeg håber ikke jeg er lavpunktet. F: Jeg har lovet ikke at sige noget til nogen. Jeg har fået det, jeg skulle bruge. Om ikke andet så får du en kopi af cheat sheet. I: Jeg sidder jo gerne og har fingrene i det, men vi skal have de andre i gang med at have fingrene i det. *Interview er slut.* *Der følger en kort snak om online kurses vedr. SAP HANA SQL Script på internettet.*	

Resulterende kode:

```
CREATE FUNCTION lbs_to_kg (in_lbs DECIMAL (10,2))
RETURNS ex_kg  DECIMAL (10,2) AS
BEGIN
ex_kg := :in_lbs / 2,2046;
END;


CREATE FUNCTION timeloen_korr(in_kat VARCHAR (30), in_timeloen INT)
RETURNS ex_timeloen_kor  DECIMAL (10,2) AS
BEGIN
IF :in_kat  = 'Civil' THEN
ex_timeloen_kor  :=  :in_timeloen * 1,6 ;
ELSEIF :in_kat  = 'Militær' THEN
ex_timeloen_kor  :=  :in_timeloen * 1,4;
ELSE
ex_timeloen_kor  :=  :in_timeloen ;
END IF;
END;
CREATE FUNCTION falkutet(in_x INT)
RETURNS ex_falkutet INT AS
BEGIN
ex_falkutet :=  1;
WHILE :x > 1 DO
ex_falkutet := ex_falkutet * in_x;
in_x := :in_x - 1;
END WHILE;
END;

FUNCTION "FORSVARET"."FORSVARET_PAC::PER_TABEL"
(in_titel varchar (30))
```

```
RETURNS TABLE (Navn varchar (30), Kategori varchar (30), Timeloen INT)
LANGUAGE SQLSCRIPT
SQL SECURITY INVOKER AS
BEGIN
RETURN SELECT Navn, Kategori, Timeloen
FROM "FORSVARET"."Personel"
WHERE Titel = :in_titel;
END;

CREATE PROCEDURE random (
IN antal_row INT, lower_val INT, upper_val INT )
LANGUAGE SQLSCRIPT
SQL SECURITY INVOKER AS
BEGIN
declare i INT DEFAULT 1;
FOR i IN 1 .. :antal_row DO
INSERT INTO RandomTable(RandomNumber)
VALUES (RANDOM()* (upper_val - lower_val) + lower_val );
END FOR;
END;
```

Transskribering af forsøg med IBM medarbejder nummer 2.

Fil: 2019-09-18_14-12-03 full.mp4

Gennemført 2019-09-18.

Tid	Tale	Resultat og bemærkninger
0:10	F: De andre, der har været her er gået smilende herfra. Det gør ikke ondt. Det her er en af de slides, jeg havde med til intro mødet. Det tager en times tid det her. Afhængigt af hvordan det går, og hvor meget jeg når at se undervejs. Om lidt så får du cheat sheet her, og det får du lige lov til at sidde og hygge dig med. Du skal ikke læse det i detaljer, men orientere dig om, hvad det dækker, og hvad står hvorhenne. Du beholder det bagefter, jeg tager det ikke fra dig. Prøv at få en nogenlunde fornemmelse, hvad der står. Når du siger, at nu har du nogenlunde overblik over det, så får du et opgaveark med fem opgaver, det er ikke sikkert, at du når dem alle sammen. Det gør ikke så meget. Det værste, der egentligt kan ske, at du løser alle opgaver lynhurtigt uden fejl, fordi det jeg har gjort, da jeg kiggede ind i sproget, var at finde nogle andre analysemodeller og fundet nogle ting, hvor man kan sige, at det her er lidt svært, og jeg har en idé om, at den model vi prøver nu, der kan jeg måske nå de samme resultater. Så hvis du kører lige igennem uden fejl, så duede den idé måske ikke. Jeg må så også sige, at de andre også har lavet fejl. I: Jeg har ikke den store programmeringsbaggrund, så jeg tror det passer. F: Det gør ikke spor, at der er de forskellige yderpunkter. Snagit har jeg aktiveret, det kører i baggrunden. Tag det bare med ro, prøv at tænke højt undervejs. Du skal nok få fred og ro til at læse det der. Og ellers så sætter jeg mig her ved siden af. Bare sig hvad du vil gøre, og hvad du tænker. Nogle gange så peger jeg på skærmen og spørger måske hvorfor gør du sådan, hvad tænkte du der. Kunne det have været løst på en anden måde. Så får vi en snak om det. Der er ikke noget, der rigtigt eller forkert, jeg prøver at forstå den proces, der foregår inde hos dig. Når vi så er færdige, så tager vi en snak om hvor meget erfaring du har. Vi taler også om, hvad du synes var nemt, og hvad du synes var svært, det kan også give mig en idé om, om jeg har ramt rigtigt, med nogle af de andre analyser. Jeg ved ikke om det kommer som en overraskelse, jeg sagde det ikke til vores indledende møde, men sproget er SAP HANA SQL Scripting. Så hvis du har taget SAP HANA 150, som nogle af dine kollegaer har hygget sig med på et tidspunkt…	I virker lidt nervøs og undskyldende i forhold til manglende erfaring som programmør. I tør lidt op undervejs.

Tid	Tale	Resultat og bemærkninger
	I: Nej, jeg har kigget på ABAP, det er det, jeg støder på lige nu. F: Det er helt fint. Prøv lige at se på det der, så om fem minutter, så kan vi prøve at snakke opgave 1 igennem. I: Okay, og opgavebesvarelsen foregår bare her i Wordpad? F: Ja det er helt banalt. Det er det ultimative udviklingsværktøj.	
08:15 **Opgave 1**	F: Hvad tænker du, er det helt sort? I: Nej, først og fremmest tænker jeg, hvad er det jeg kommer til at rykke rundt med? Jeg er ikke så vandt til at programmere. Der kan jeg så se, at det har noget at gøre med tabeller. Det er jo lidt… Der er måske lidt dér, hvor jeg kan blive overrasket. Som udgangspunkt tænker jeg, at det er noget data, som jeg måske skal manipulere. F: Lad os prøve at kigge på det. Her er opgavearket, det første heroppe er en generel tabel, som går igen i nogle af opgaverne, men du skal ikke bruge den i opgave 1. Så den kan du vente med at kigge på til opgave 2. Så er det sådan, at de første opgaver, da er det egentligt et spørgsmål om at finde det rigtige sted i cheat sheet, og så har du skelettet. Så skal du tænke over, hvordan du bruger det, så det kommer til at passe. Så bliver det lidt sværere, i takt med at du kommer igennem det, og så ser vi hvor langt du når. Så prøv at kigge opgave 1 igennem, når du har læst den, kan vi tage en snak om, hvad det går ud på. I: Jo, her der tror jeg, man får en værdi, som er vægt i noget, og man vil gerne have et output, som er i en bestemt enhed. Så jeg forestiller mig en if-rutine, kan anvendes her, så man kan få det enten i kilo eller i pund. Og her vil jeg antage, nej det står specifikt at output skal være i kilo. F: Der står faktisk også at input er i pund, så du behøver ikke lave en if. Du ved, at du får et input. Du skal lave en funktion, hvor du får noget ind i pund, og du skal levere det ud i kilogram. I: Altid pund, så skal det bare omregnes. F: Det næste er at finde et sted, hvor du har en funktion, der returnerer noget. Hvis du i cheat sheet kan finde en	F hjælper og vejleder lidt mere end hos de øvrige respondenter for at sikre en god start og prøve at fjerne den indledende nervøsitet. Bruger undervejs punktum efter END. Sletter det efterfølgende men tilføjer ikke semikolon.

Tid	Tale	Resultat og bemærkninger
	funktion, der returnerer et tal, så vil du være hjulpet. I: (Læser og mumler). Der er en skalarfunktion her. F: Lige præcis, en funktion som har… I: Returns a single numeric type. De limitations der er, stemmer med det vi har såhh… jo, så ville jeg nok skrive noget, der minder om det her. F: Det må du meget gerne kaste dig ud i. I bund og grund få de fem linjer der til at ligne noget, som kunne få en vægt ind i pund og spytte den ud i kilo. I: Jeg er lidt usikker på om det der UDF… F: Det er bare en kommentar. I: Hvis vi nu siger vi har en kommentar, konverter l b s til kilo. (Skriver og mumler). Her er jeg i tvivl om, hvad i m betyder? F: I m er en notationsform, der indikerer at her er det et input. Nogle steder står der out eller e x for det der leveres ud. Ligesom hernede, her kalder de den i m for input parameter og e x for det, der kommer ud af det. Det er en notation SAP bruger – det er ikke afgørende. I: Her vil jeg kalde den pounds, så står der INTEGER, så tænker jeg umiddelbart, at det er noget med datatyper. Her kan vi have et kommatal, så jeg skal nok. INTEGER er hele tal, så jeg skal finde noget med decimaler. Total og fraction, når der står t f her, er det hvor meget der kan stå før kommaet og efter kommaet? F: Nej, total er antal cifre i alt, og fraction, er hvor mange du har på højre side af kommaet. I: Okay, pounds det kan jo være…, jeg ved ikke om det er nødvendigt at specificere, men jeg vil jo nok gøre det. DECIMAL og så parentes total fem, og så tre, og en parentes mere. RETURNS den vil jeg så kalde e x k g i stedet for result. Det er en DEC, skal vi beholde det samme, så siger vi AS, BEGIN, ex k g, (skriver og mumler). En tanke jeg har nu, er om jeg skal kalde det ex result eller ex k g, som jeg har her. F: Det der er rigtigt og vil kompilere fint. Jeg er ret sikker på, at hvis vi havde HANA Studio liggende, så ville det kompilere og virke fint. Du må meget gerne hoppe videre til opgave 2.	

Tid	Tale	Resultat og bemærkninger
19:30 **Opgave 2**	I: (Læser) Okay, her er det lidt det samme, tænker jeg. Jeg har noget input, og output skal afspejle hvorvidt, der er tale om noget overarbejde, så jeg tænker, at det er lidt det samme, men fordi jeg har flere muligheder, så tænker jeg, at det eksempel, der er hernede er meget interessant at kigge på. Det er en IF rutine, en tanke jeg gør mig, er at når man laver noget med en IF, så er det enten eller, dvs. man har to muligheder for et output. Her har vi tre, hvis man er militær så bliver det ganget med 1 komma 4, hvis kategorien er civil bliver det ganget med 1 komma 6, hvis det er hverken eller så er det uændret. Det jeg kan se ud fra det her er, at der er en IF, så er der en ELSEIF og så er der en ENDIF. Der er tre udfald, så jeg vil gå videre med den her. F: Det er fint. I: (Skriver og mumler) Det jeg tænker lidt over, er hvad er det, jeg gerne vil have regnet ud. Det er lønnen, så jeg kalder den bare kalk, det skal være med småt. Grunden til at jeg tror, at det skal være med småt er, at det kan jeg se i eksemplet. F: HANA er faktisk ikke kritisk med det. I: Når jeg så har navngivet min funktion, så skal jeg have et input på samme måde som før. Her er mit input, jeg har to input: kategori og timeløn, så jeg er lidt i tvivl om, hvordan jeg får dem begge to ind, eller om det er nok med en, som udgangspunkt vil jeg sige, at…, jeg vil have timeløn som en inputparameter, nej (skriver og mumler), så skal jeg definere datatypen, og der vil jeg nok sige decimal. F: Her får du faktisk lidt hjælp fra tabellen, som jeg sagde, at du først skulle bruge i opgave 2. Her kan du se, den er sat på som INTEGER, og at kategorien er en VARCHAR på 30 karakterer. I: Ja, timeløn skal være INTEGER, og når det er en INTEGER skal man så definerer antal tegn. F: Nej, det skal man ikke. I: (Skriver) Den anden parameter, det var kategori, den kalder vi cat, og den skal så være VARCHAR, det kan jeg se ud fra tabellen, og hvor mange tegn – 30, så RETURNS, hvad vil jeg gerne have den til at returnere, det er vel bare løn, det skal være, det er nok kommatal fordi vi ganger med 1	I er usikker i forhold til logikken. I er usikker i forhold til at skrive en rutine, som kan behandle to inputparametre. I er usikker i forhold til, hvad en INTEGER kan indeholde. I forsøger at finde ”mønstre” i koden i stedet for at klarlægge syntaks. I er noget usikker på forgrening ved hjælp af IF-ELSEIF-ELSE-ENDIF.

Tid	Tale	Resultat og bemærkninger
	komma 4 og 1 komma 6, DECIMAL, tegn i alt, vi siger bare seks komma 2, det er stort. F: Njah, det er en fire cifret timeløn og to decimaler til ørene. Det er meget fint. I: As, så kan vi komme til min IF. Det jeg skriver nu, er hvad det er, der gør om jeg skal gange med 1 komma 4 eller 1 komma 6, så det er kategorien, der er afgørende. Så jeg vil skrive navnet på min kategori, som jeg har sat til i m underscore cat. Og jeg havde to kategorier, militær og civil, nej egentligt tre, for det kan være, at der ikke er noget. Jeg starter med militær og (skriver og mumler). Det er mit bud. Jeg kunne lige kigge det igennem (læser og retter). Min hjerne ser en slags mønster i forhold til store og små bogstaver, tegnsætning i forhold til semikolon, hvor jeg tænker, at når koden gør noget nyt så skriver man semikolon, så er der en ENDIF, når jeg tænker mere over det, så er jeg ikke sikker på, om den lukker ELSEIF, og END så lukker for IF, eller hvordan eller hvorledes. Men jeg føler alligevel, at det er sådan her, det skal skrives, selvom jeg ikke er helt sikker. Skal jeg hoppe videre til den næste opgave? F: Det må du gerne. Jeg tror, at opgave 2 vil kunne køre. Der er lige et par slå fejl. Du behøver ikke lede efter dem. I: Jeg tænker, at compileren vil klare det.	
33:20 **Opgave 3.**	I: Den her den er måske en lille smule tricky. Nu skal jeg over og kigge i cheat sheet. Jeg forstår det sådan, at hvis der fx stod 10 i stedet for 5, altså 10 og så !, så har du 10 gange 9 gang 8 og så hele vejen ned til 1 og så skal den returnere resultatet af det. F: Ja, udråbstegnet er bare den matematiske måde at skrive fakultet på. Det er ikke noget du kommer til at bruge i koden, det udråbstegn. Du skal skrive en funktion, der tager en værdi ind, og så skal den levere den værdis fakultet. I: Okay, jeg tænker, at der må være en funktion indbygget. F: Det er den ikke – desværre – du er nødt til selv at lave et eller andet, der løber nogle tal igennem, og så ganger dem sammen. I: Okay, den er jeg så lidt i tvivl om, hvordan jeg skal gribe an. For det kan jo være, jeg tænker på muligheden af	F vejleder og støtter noget mere end hos de øvrige respondenter. I har ikke tilstrækkelig grundlæggende viden om imperativ programmering til at kunne opstille logikken, der er nødvendig for at kunne løse opgaven. Forklaringen af forskellen på = og := er usammenhængende og giver ikke entydig mening.

Tid	Tale	Resultat og bemærkninger
	kombinationer, og hvordan jeg får det repræsenteret i kode. Det er måske bare noget med minus en, og så skal jeg have den til at stoppe, før den rammer nul. Altså jeg tænker umiddelbart, jeg ved ikke lige hvilken del af cheat sheet, der vil være brugbart her. F: Du skal nok have noget, der looper tallene igennem. I: Altså noget imperativ programmering. F: Ja, du kan både bruge en FOR loop, du kan også bruge WHILE. I: (Læser) Jeg kan godt lide den her del, hvor der står processing and exit loop, hvor man sætter x til at være…, det ville jeg gøre med tallet et, så ligeså snart du når tallet et, så skal den stoppe, den skal ikke fortsætte ned til nul, men er alt det her en? F: Nej, det er faktisk to forskellig metoder. Umiddelbart ser det nederste nemmest ud, men det øverste er faktisk nok det de fleste har nemmest ved at forstå. Der er så lidt med CONTINUE og BREAK, der er sat ind, for at gøre eksemplet komplet, men det er faktisk ikke nødvendigt. Jeg tror, at du skal starte med lige at lave et skelet, hvor du bare lige har nogle informationer, og siger, hvad den skal returnere og så en END, så kan vi tale lidt om, hvad vi skal have fyldt ind imellem. I: På samme måde som i opgave et. F: Ja, ligesom opgave 1, du får en værdi ind og den er så den, du skal finde fakulteten af. I: (Skriver og mumler). Hvordan looper den, jeg vil gerne have den til at tage min x og så gange med x værdien minus en, resultatet af det bliver så ex_faculty, men nedenunder har jeg skrevet IF x større end et, så skal den fortsætte, men måske skal det være, IF ex_faculty større end et, så skal den fortsætte, fordi resultatet af det her vil være ex_faculty, resultatet af x gange x minus et, har jeg defineret til at være ex_faculty, så er jeg lidt i tvivl om, om jeg skal fortsætte eller skrive faculty her, nej det skal jeg ikke. Nu er jeg i tvivl, jeg kan ikke helt se, hvordan jeg får x til at…, den skal så tage fire næste gang, mens den er fem, det ved jeg ikke helt, hvordan jeg får inkorporeret. F: Det betyder ikke så meget for opgaven her, men der er	

Tid	Tale	Resultat og bemærkninger
	nogle andre ting, hvis du kigger her, nogle gange har du et lighedstegn, og nogle gange kolon lighedstegn. Hvad tænker du er forskellen på at skrive lig med og kolon lig med? I: Jeg checker lige med mit cheat sheet. For jeg har en tanke, men jeg skal lige have den bekræftet. Jeg tror, at når du bare har lig med, så definerer du, eller du forklarer i koden, at hvis du fx har kategorien militær, så derfor så starter med kolon foran det du har defineret det som, og så har du bare et lig med uden nogen tegn…(gætter). F: Godt, lad os prøve at hoppe videre til opgave 4.	
45:50 **Opgave 4**	I: (Læser og mumler). Det man gerne vil have her, er en oversigt over nogen, som har en bestemt titel i en tabel. Den refererer så til den tabel 1. F: Hvis du kigger herovre, så har du fx titlen her. Hvis vi nu lavede et udtræk, hvor vi gerne vil have alle dem, der har titlen kaptajn, så er det Ole Olsen og Niels Nielsen. For de der har den titel - kaptajn, vil vi gerne se navn, kategori og timeløn. Du skal have skrevet en rutine, som spytter en tabel ud. I: Der tænker jeg umiddelbart, at det er noget med SELECT, nej lad mig lige se i cheat sheet. F: Det er ikke helt tosset, du er bare et skridt længere fremme. Du skal nok starte med at finde dig et skelet, du kan hænge det op på. I: (Læser) I det her tilfælde ville jeg nok skrive SELECT titel og så fra tabelnavn, som hedder personel, WHERE category, jeg tænker, at der skal stå noget med nogle feltnavne altså kolonnenavne her, jeg ville nok gøre noget i den stil. F: Ja, hvad var det outputtet skulle være? I: Outputtet skulle være navn, kategori og timeløn. F: Og det skal komme ud i en…? I: I en outputtabel. F: Lige præcis. Du kan se den her, den returnerer netop en table, så det er hele skellettet her, du skal have fat i, og så skal vi om lidt til at tænke lidt på, hvad det er for en logik, der skal være. Men du skal have forberedt en funktion, der	F vejleder og støtter noget mere end hos de øvrige respondenter. I er hurtig til at pege på SQL. Det indikerer, at I er mere rutineret i brug af SQL sammenlignet med imperativ programmering. I virker ikke rutineret i brugen af HANA database teknologi, herunder forståelse for brugen af schema og repository. I støtter sig i høj grad til koden i cheat sheet.

Tid	Tale	Resultat og bemærkninger
	returnerer en tabel. Og så er der noget med, at du skal definere hvilket sprog, det kører i. Så BEGIN og END, og så kommer logikken herinde. Der er lidt skrivearbejde. I: (Skriver og mumler) F: Den er lidt svær den her. Det var det jeg sagde om, at sværhedsgraden stiger undervejs. I: (Læser) Det er jeg ikke sikker på, at jeg forstår. F: Jeg ved ikke, hvor meget HANA, du kan. Det her er HANA tankegang. Det her er kode, vi skubber ned i databasen. En HANA database, er et database hotel, som kan indeholde mange databaser. En database ligger i et schema. Så der ligger i forvejen et skema, der hedder FORSVARET nede i det her databasehotel, og det er det, vi arbejder med. Sammen med schemaet ligger der også et repository, hvor vi kan placere vores kode. Og det er også defineret i forvejen, det hedder forsvaret underscore pac. Hvis du hopper tilbage til cheat sheet, så kan du se, her har du først schemaet, det var FORSVARET, og så hed det FORSVARET underscore PAC for det repository, du placerer din kode i. Så kan du begynde at definere dine input, og så skal du returnere en tabel, og der skal du så definere, hvad der skal komme ud af den funktion. Lad os tage det stille og roligt. Først skal du have defineret schemaet. I: Ja det står her, det var FORSVARET. Og så FORSVARET underscore PAC. (Skriver og tænker højt, mindre drøftelser undervejs). F: Lad os lige tage det, du har skrevet. Du siger, at den skal tage navn, kategori og timeløn der, og det skal den tage fra schemaet FORSVARET, så nu ved vi hvor vi er henne. Så skal vi have fundet tabelnavnet. Det her er et kolonnenavn. Vi skal have fundet ud af… hvad hedder den tabel, som du skal hive de tre informationer ud af? I: Personel tabel. F: Prøv at se, heroppe har du valgt at kalde din inputparameter for titel, hvis du beholder den samme notationsform og så skriver i m foran, så får du skilt ad, hvad er inputvariabler, og hvad er kolonnenavn, så det er der, hvor titlen er lig med input, kan du følge mig. I: Ja, (skriver). Så var der noget med kolon, jeg tror, at jeg	

Tid	Tale	Resultat og bemærkninger
	sætter kolon her foran, og så semikolon til sidst. Det minder om, det er jo SQL et eller andet sted. Det, jeg har erfaring med, det jeg primært har arbejdet med, minder om det. F: Ja, i ABAP ligger SQL også indlejret som ABAP SQL. Det kunne jeg også godt se, da du begyndte at skrive, at her var du mere på hjemmebane. Nu er der gået en time.	
59:20 Afslutning	F: Jeg skal høre dig ad, hvad du synes har været nemt, og hvad har været svært? I: Før jeg startede hos IBM, havde jeg ingen programmeringsbaggrund. Jeg synes, at fx når du peger mig i en retning, så hjælper det rigtigt meget. Det hjælper også meget at have et cheat sheet. Ekstremt meget. Jeg tror, at jeg skal finde et cheat sheet til ABAP. F: Jeg kan sige til dig, at der ikke fandtes et til HANA SQL Script, så jeg har måttet lave det her. Jeg vil tro, at ABAP er så udbredt, så det vil undre mig, hvis der ikke findes et. I: Det hjælper utroligt meget at have et cheat sheet, med nogle forholdsvis simple eksempler. Det hele handler om at komme ind i tankegangen, man skal lige forstå logikken og samspillet mellem de parametre, man definerer, og det man gerne vil have ud. Hvis jeg ikke har det overblik, over hvad det er jeg forsøger at gøre, så kan det godt være lidt tricky. Det hjælper også, at opgaverne er klaret defineret, der får at vide, hvor du skal slå op og kigge henne. Den information er jo ikke altid noget, vi har lige ved hånden, men det er jo bare en del af virkeligheden og hverdagen. Når du ved, at du skal slå op et sted, altså hvis du kender funktionen eller det, du prøver at skrive, så vil du automatisk tænke i den bane, fremfor bare at taste derud af. Det var en stor hjælp. F: Prøv at kigge i cheat sheet omme på den anden side. Så er det mit indtryk, at punkt 3 – det med at loop – der var du lidt famlende. I: Altså loop, det har jeg stadigt ikke…, jeg forstår at du tager en record, og så behandler du den, så kører du igen, men hvornår stopper det, hvor starter du, hvor mange gange looper den, i min dagligdag, så hører jeg, at så looper den én gang, men er det en standard ting, at den looper én gang. Det er sådan nogle ting, jeg ikke lige helt ved, hvordan jeg skal tage højde for.	

Tid	Tale	Resultat og bemærkninger
	F: Jeg kan også sige, at den kode du har skrevet i opgave 3, den vil ikke kunne køre. Selvom den kunne køre, så ville den ikke give det resultat, den skulle. Men jeg kunne høre, at du havde tankegange, med hvordan får jeg den til hele tiden at blive ved, indtil den når ned til en. For bare lige at springe til det. Så den der linje FOR, der fortæller man den, at x skal antage alle værdi mellem 0 og 10 i dette tilfælde. Første gang man løber igennem, så siger x, at nu er jeg nul, og kører den det her igennem, indtil den støder på END FOR, så hopper den tilbage, så nu er den en, så kører alt det her igennem, tilbage igen nu er den to, det bliver den ved med, indtil den når det her tal. I: Jeg ville jo have den til at køre op til uendeligt, fordi jeg ikke har nogen grænse. F: Du skal jo kun lade den køre indtil, det tal, du har her. Fidusen er at fem gange fire gange tre gange to gange 1 er det samme som en gange to gange tre gange fire gange fem. Du sad og tænkte i at tælle baglæns, kunne jeg høre. Men der er noget, du skal have sat dig ind i, når du kommer dertil. I: Det er rigtigt set, det var svært. F: Jeg tror, bortset fra opgave 3, så vil de andre kunne køre, det kan godt være at der lige mangler et semikolon et sted. Men hvis du arbejde i HANA Studio, så ville du få de warnings undervejs. I: Sidst jeg var i HANA Studio, var det forholdsvist simpelt, men du ved, det vindue der er, der er også hjælp at hente, så man kan komme ind på sporet. Så ved jeg ikke, om der er en F1 funktion ligesom i det gamle (SAP GUI), men det må der næsten være. F: Der er i hvert fald noget hjælp. Men om det kræver, at man er på internettet, fordi man går ud og henter sider, det kan jeg ikke huske. Da jeg arbejdede med HANA Studio derhjemme for at lave de her opgaver og teste dem, der brugte jeg i hvert fald F1. I: Den her måde at gøre det på, er også meget lærerig. Jeg føler, at jeg er blevet klogere på den her time. F: Okay, det er fedt. Jeg har også fået fine data samlet ind. I: Nu ved jeg ikke, hvis man i en virksomhed skal skifte til	

Tid	Tale	Resultat og bemærkninger
	noget andet, tænker man, at det er for stor en omkostning, men det giver meget. F: Man kan sige, at i det her tilfælde, hvis man vil skubbe kode ned i HANA databasen for at opnå de mulige hastighedsgevinster, så er der ikke andre muligheder end at skrive det i SAP HANA SQL Scripting, så der er ikke noget valg. Så kan man sige, hvor stor er omkostningen, hvis man gør det. Det kan være interessant at få belyst. Et udfald kunne være, at vi sagde til alle fra IBM, nu skal i lave maksimal code pushdown til HANA databasen, og det kan I lære på en dag eller to, men det kunne også være, at vi skulle have Frozen Zone i tre måneder, fordi I ikke kunne lave andet end at sidde på skolebænken, og det er de yderpunkter, som kan være relevante at finde estimater på. I: Det med tid, du vil finde ud hvor lang tid. F: Jeg har analyseret sproget med nogle andre værktøjer, og så vil jeg gerne se, i stedet for at bruge de metoder, bare tog jer alle sammen og brugte en time af jeres tid, hvor meget klogere, kan man så blive på, hvor meget der skal til for at omstille jer. Giver det de samme resultater, som de store forkromede løsninger, og det ved jeg ikke endnu. Det kan være, når jeg har haft jeg igennem alle sammen, at jeg begynder at se et mønster, og kan sige, at det I syntes var svært var også det de andre metoder viste. Eller det der ligger lige til højrebenet, var også det de andre metoder viste, lå lige til højrebenet. Jeg kunne se på dig, at det at kode SQL inde i noget andet kode, forekom meget naturligt. I: Det er jo det, jeg har øvet mig på – SQL, jeg får meget af det. F: Forskellen er, at du får de muligheder herovre, som IF-THEN, ELSE, du kan lave loops, du kan sætte variabelnavne til at være lig med en tabel, det kan man ikke i SQL, der er nogle udbygninger i SQL, som gør det mere fleksibelt, men det kræver så også, at man har overblik over det. Der var en enkelt ting, som jeg glemte at spørge dig om i opgave 3. Som jo ikke kørte helt. Der står, at x fakultet kan blive et meget stort helt tal. Kan du huske, at du satte det til at være en INTEGER. I: Nåh ja, jeg tænkte på hvor mange tegn, der skal angives. F: En integer fylder 32 bit, det vil sige, der går en bit fra til fortegn, så det er 31 bit. Det er et stort tal, men ikke et	

Tid	Tale	Resultat og bemærkninger
	særligt stort tal. Spørgsmålet er, om… I: Det ved jeg ikke så meget om. I mit hoved, så kan det gå til uendeligt. F: En INTEGER er bestemt ikke uendelig, hvorimod en BIGINT er betragteligt meget større. I: Jeg kom bare ind i en tankegang om, at det kunne blive uendeligt, men det er selvfølgeligt rigtigt, at på et tidspunkt så er der ikke længere plads.	

Resulterende kode:

```
Opgave 1
*Konverter lbs (punds) til kg


CREATE FUNCTION convert_pounds (im_pounds DEC (9,3))
RETURNS ex_kg DEC(9,3) AS
BEGIN
          ex_kg :=:im_pounds / 2,2046;
END


Opgave 2
CREATE FUNCTION kalk_loen (im_loen INT, im_cat VARCHAR(30))
RETURNS ex_loen DEC(6,2) AS
BEGIN
          IF :im_cat = 'militær' THEN
          ex_loen := :im_loen * 1,4;
          ELSEIF :im_cat = 'civil' THEN
          ex_loen := :im_loen * 1,6;
          ELSE
          ex_loen := :im_loen;
          END IF;
          END;
```

Opgave 3

```
CREATE FUNCTION calc_faculty (im_x INT)
RETURNS ex_faculty INT AS
BEGIN
ex_faculty :=: im_x * (im_x - 1);
IF :ex_faculty >1 THEN
CONTINUE;
END IF.
```

Opgave 4

```
FUNCTION "FORSVARET". "FORSVARET_PAC: :titel_tab"
                                                    (im_titel VARCHAR(30))
RETURNS TABLE (navn VARCHAR(30),
                      kategori VARCHAR(30),
                      Timeloen INT)
LANGUAGE SQLSCRIPT
SQL SECURITY INVOKER AS
BEGIN
           RETURN
           SELECT navn, kategori, timeloen
           FROM "FORSVARET". "personel"
WHERE titel = :im_titel;
END;
```

Transskribering af forsøg med IBM medarbejder nummer 03.

Fil: 2019-09-24_10-34-33.mp4

Gennemført 2019-09-24.

Tid	Tale	Resultat og bemærkninger
02:06 Indledning.	F: Har de andre sagt lidt om, hvad det går ud på? I: Jeg ved ikke mere end det du sagde den dag. Det er nok ligesom at lære at køre bil, der er bedst ikke at vide for meget i forvejen. F: Jeg tror nu at de har fundet det sjovt at være med i. Det er stille og roligt, det er jo ikke jer, der bliver testet. Det er metoden. Det her er en af de slides, jeg havde med. Om lidt går vi i gang, og så skal du regne med at den tid, hvor du skal være aktiv tager en times tid, og så er der lidt snak før og efter, så om halvanden time burde vi være færdige. Lige om lidt så får du det her cheat sheet, og det giver jeg dig så en fire til fem minutter til at læse igennem. Du skal ikke sætte dig så meget ind i, hvad der står, men mere hvor kan du finde hvad. Kig på overskrifter og læs det, uden at gå i detaljer. Jeg ved ikke om du har været vandt til at arbejde med cheat sheets. I: Næh, mit cheat sheet hedder google. F: Det er også godt. Det får du fred og ro til at læse, og når du har gjort det, så får du opgavesættet her. Der er fem opgaver i det. De første er relativt tilgængelige, så der er det et spørgsmål om at finde det sted i cheat sheetet, hvor du kan finde skabelonen, og så få den tilpasset så det virker. Jeg fik vist ikke sagt, at det er HANA SQL Scripts, vi arbejder med. Så vis du har taget HANA150… I: Jeg har læst det igennem, men jeg har ikke brugt det, og det ligger langt tilbage. F: Det er der også andre, der har sagt. Men det kan være, der er noget der vækker genkendelsens glæde – det må vi se. Og så bliver sværhedsgraden lidt sværere, men det kan vi komme tilbage til. Jeg har ingen forventning om, at du kommer helt igennem. Det værste der kan ske, set fra min side, set fra min stol, det er at du bare løber det hele igennem, og laver alle opgaverne uden at lave fejl undervejs, for jeg vil gerne spotte, om det der er svært ved dette sprog, også er det jeg har fundet med andre analysemodeller. Det	I virker afslappet og åben i forhold til forsøget. I har læst kursusmaterialet fra SAP HANA150, men har ikke brugt det. Det var som det fremgår ikke intentionen at transskribere samtlige forsøg. Det viste sig dog efterfølgende at være givtigt.

Tid	Tale	Resultat og bemærkninger
	at tage jer ind i kun et time skulle helst ligne det fra de andre måder, men det ved jeg ikke endnu, det kan også være at resultatet bliver anderledes. I: Så du prøver at holde cheat sheet op imod alternativer. F: Lige præcis, jeg har forsøgt med andre modeller til at analysere SQL Scripting ud fra og fundet nogle ting, der er nemt og svært ved det her sprog, og nu prøver jeg at kaste jer ud i det, for at se om det er det samme, som I oplever er nemt og svært, det samme der kan drille lidt – så det gør vi. Du må meget gerne tænke højt, de andre har været gode til at sige: her er en type problem hvor jeg skal have fået i noget i denne stil, jeg kunne gøre sådan, eller jeg kunne også gøre sådan, og så snakker vi om det under vejs. Det kan være jeg siger, at du skal kigge herover, eller jeg siger at det ikke har den store betydning. På et eller andet tidspunkt siger jeg, at nu er vi færdige med opgave et, det er ikke sikkert at, den er helt rigtig, men der er kommet det ud af det, som jeg gerne vil se. Snagit er sat op, så det kører i baggrunden, det skal du ikke tænke på, men det betyder, at alt det der foregår, bliver recorderet inkl. lyd. Jeg har ikke en forventning om, at du bliver færdig, så tag det med ro. I: Så det interessante er, om man kan bruge cheat sheet. F: Ja, eller det, der er interessant er egentligt, hvad får man ud af kun at bruge en time af din tid, ved at give dig et cheat sheet og nogle opgaver, og så holde øje med, hvad der er nemt eller svært. Når vi er færdige har jeg nogle spørgsmål, der går på, hvad du syntes var nemt eller svært, og så ganske kort om sproget ligner noget, du har prøvet før, vi har talt om, at du har læst lidt om det, så vi taler om hvor meget erfaring du har. Er du frisk. I: Det kan du tro. F: Så får du den her og fem minutter. Der er noget på begge sider.	
12:19	F: Hvad siger du om det? I: Det ser spændende ud. Det er jo nok niveauet. Et cheat sheet kan jo ikke blive en komplet manual.	

Tid	Tale	Resultat og bemærkninger
	Nej det bliver det ikke. Her er opgavesættet. Det øverste heroppe er en tabel, som ligger i systemet. Her har du dit programmeringstool. Det er verdens mest avancerede tool, det er Wordpad. Du skal tænke på, at tabellen ligger inde bagved. Du skal ikke bruge den i den første opgave, vi vender tilbage til den, når det bliver relevant, jeg tror det er i opgave 2. Jeg vil foreslå, at du prøver at læse opgave 1 igennem, og så kan vi tale om hvad du skal i opgaven.	
13:30 **Opgave 1**	I: (læser) Det var den. Hvis jeg forstår det rigtigt, så skal jeg lave en rutine, der kan omregne mellem kilo og pund, og inputparameteren det er pund, og den skal levere kilo tilbage. F: Lige præcis. Så én inputparameter og én output. I: Det var opgaven, og det her omregningen. Så skal jeg har fundet min… det er noget med en inputparameter, og det var der vist en af de her, der kunne. User defined function can contain only single. Det passer næsten perfekt. Create function return begin, så skal jeg have lidt imperativ for at lave omregning. F: Prøv at starte med skallen, så kan vi se på det. I: Ja, create, convert fra pund til kilo, så vi kalder den pund til kilo og inputparameteren hedder pund, skal der type på, det skulle der, gad vide om det er en decimal, (mumler), hvor mange pund kan man forestille sig? F: Hvad synes du? I: Tusind kunne man godt forestille sig og så to decimaler for fraction. Og på den anden side. Det var inputparameter, så skal vi haven en output parameter, det var kilo, som også bliver decimaltal. RETURNS kilo. AS. Så må det være her, at logikken skal komme. F: (Peger i cheat sheet). Kigger du her? I: Ja, nu skal jeg til at lave omregningslogikken. F: Med fordel kunne du måske kigge heroppe. For hernede har du et eksempel med to inputparametre. Heroppe er	I forståer ikke definitionen af DECIMAL(x, y) tror fejlagtigt, at x er størsteværdien – ikke antal cifre. Definerer derfor PUND DECIMAL(1000, 2). Glemmer semikolon efter END.

Tid	Tale	Resultat og bemærkninger
	eksemplet med en inputparameter. Det andet kommer vi til om lidt, du skal nok få lov til at hygge dig med det. I: Okay, begin og end. Kilo kolon lig med pund divideret med 2 komma 46. Så vil jeg godt straks korrigere, for jeg kan se, at der er fire decimaler. Det vil jeg sige, at det er mit første skud på det her. Vi har lavet en funktion CREATE function Convert pund to kilo. Og inputparameteren det er så pund defineret som en DECIMAL op til 1000 med fire decimaler. Og den returnere kilo også i formatet op til 1000 også med fire decimaler, og så funktionen AS defineret her mellem begin og end, sat lig med pund divideret med 2 komma 46. F: Det kan nok virke et stykke hen ad vejen. Jeg har måske forvirret dig ved at sige at pund kunne være op til tusind. Her har du angivet, at der kan være tusind decimaler. I: Når jeg opfattede det som, hvor var det? F: Ja, antal decimaler i alt, og hvor mange af dem står så efter kommaet, så hvis du fx sagde otte decimaler og de fire af dem på højre side af kommaet. I: Ja, så tallet er totalt 8 og de fire kan være decimaler. F: Det ser fint ud, jeg syntes at vi hopper videre til opgave 2. I: Skal vi skrive opgave 1 her, så er der styr på det.	
21:00 **Opgave 2**	I: Opgave 2. Hvad skal vi der? Lad os se. Okay. Den opfatter jeg egentligt som en udbygning af den første opgave, nu har vi to inputparametre og én returparameter. Plus vi så også skal have en imperativ programmerings if-sætning ind i, for at styre om kategori er militær, civil eller om der ikke er nogen kategori. Men ellers så er selve grundskallen stadigt en funktion man skal bygge. F: Skal vi lige kigge på tabellen, som jo er input her. Man får jo en af linjerne her, der er så der, man plukker kategorien, titlen og lønnen ud. Du kan også se her, at nogle steder er den militær, nogle steder civil og nogle steder slet ikke defineret. Her i tabellen har du fået hjælp, idet datatypen er opgivet. Så det er bare at klø på. I: CREATE FUNCTION beregn timeløn, og skal jeg så bevare VARCHAR det er det samme, som bruges i inputparameter deroppe kategori, kategori som den første inputparameter	I anvender IF-ELSEIF-ELSEIF-ENDIF lidt omstændigt. Det kan indikere lidt manglende rutine i imperativ programmering. Fejl i antallet af parentes slut i forbindelse med definition af DECIMAL. Medtager ikke kolon foran TIMELON, når der refereres til indholdet af variablen.

Tid	Tale	Resultat og bemærkninger
	med datatypen VARCHAR 30, er der skilletegn mellem parametre, ja det er komma, den kalder vi bare timeløn, så behøver vi ikke mere, vi skriver bare integer. F: Der er valgfrihed i SQL Script. Du kan både skrive INT og INTEGER. I: Og så skal vi have en returparameter, som er RETURN, så den kalder vi bare beregnet timeløn AS, formatet på den, det er også en INTEGER, eller også skal det være – det bliver INTEGER. AS. F: Det er ikke så afgørende, med det skal nok være en decimal som output. Hvis der er noget folk er meget nøjeregnende med, så er de ørene på deres lønseddel. I: Decimal fem plus syv komma to, der vel ikke nogen der får 100.000 i eksemplet. F: Nej, ikke i timeløn. I: Så skal vi ned til selve funktionen. Kolon, tilgå, kategori lig, har vi ELSEIF, det har vi, THEN, militær. Vi ser bort fra at den værdi kan komme med både stort og småt. Punktum til sidst, nej semikolon. Hvis kategorien er militær, så skal timelønnen ganges med en komma fire. Hvis den er civil gange en komma seks, hvis den er uspecificeret, så er det bare timelønnen lig med inputparameteren. Så skal vi lige kigge det igennem. Vi har lavet en ny funktion, beregn timeløn, vi har to inputparametre, kategori og timeløn INTEGER, og den returnerer en beregnet timeløn i decimal. Der kan være op til syv decimaler, hvoraf to er efter kommaet. Og logikken der bruger man så den ene inputparameter kategorien til ligesom at styre if-sætningen ud fra om værdien har militær, civil eller blank, ud fra det beregner man beregnet timeløn ud fra inputparameteren timeløn ganget med satsen, der er oplyst i opgaven, som er en komma fire og en komma seks, eller at man bare sætter beregnet timeløn til timeløn, hvis man ikke har fået oplyst en kategori. Så tror jeg faktisk, at vi er ved at være der. F: Der er lige lidt med parenteserne i linje to. I: Ja, det kan jeg godt se. F: Så har jeg et spørgsmål. Du starter med en IF, så har du en	

Tid	Tale	Resultat og bemærkninger
	ELSEIF, og en ELSEIF og en ENDIF. Det kan fint køre, men det kan også gøres på en anden måde. Kan du gennemskue det? I: Ja, man kan måske undlade ELSEIF hernede, så det bliver alt andet. Så kunne man nøjes med ELSE. F: Så skal jeg høre dig ad, heroppe sætter du kolon foran kategori, hvorfor gør du det? I: Der stod et eller andet sted i cheat sheet, at man kan tilgå variable ved at at bruge kolon. F: Men kan du så forklare mig, hvorfor der ikke er kolon foran timeløn? I: Nej, det kan jeg ikke – men det burde der måske være. F: Ja, præcis. Kan du forklare mig, hvad der er forskel på kolon lig med du bruger her, og lig med du kun bruger her. I: Her der spørger man på værdien og her sætter man den lig med. F: Ja super, lad os hoppe videre til opgave 3.	
33:02 **Opgave 3**	I: Opgave 3. (læser opgaven højt). Herovre på mit cheat sheet har jeg egentligt den funktion? F: Det kan jeg godt afsløre, at du ikke har. Her skal du selv i gang. I: Det må være noget med at loope, man får en værdi ind og så skal man loope på en eller anden måde. Okay, det er stadig en funktion med en inputparameter og en outputparameter. Function, vi kalder den opløft, og vi får en inputparameter som er en INTEGER, n som INT, og den returns, beregnet n, det bliver et meget stort tal, BIGINT, når man skriver den behøver man ikke skrive hvor lang den skal være, begin, AS, øhhh. F: Du sagde et ord før? I: Loop, men jeg tænker på, om jeg har brug for en mellemvariabel, det har jeg vel ikke, den kan god finde ud af at loope fra n ned til en, det er jo det jeg ønsker, så hvis jeg taster fem ind, skal den loope fra fem ned til en, så skal jeg have en beregnet værdi pr. loop, som jeg skal tilføje en totalværdi og så returnere til sidst.	I er usikker på at udarbejde logikken i funktionen. Det kan indikere lidt manglende rutine i imperativ programmering. I har problemer med kolon foran variabelnavne, når der refereres til indholdet. Glemmer semikolon efter END.

Tid	Tale	Resultat og bemærkninger
	F: Det kan gøres på mange måder, så jeg siger ikke så meget, jeg hører blot hvordan du resonerer. I: Godt, hvor har vi loops. FOR n loop x of times. Kan man sætte den til at loope baglæns, det er vel egentligt også ligegyldigt, om man kører den ene eller anden vej. F: Det du siger er at: fem gange fire gange tre gange to gange en det er det samme som en gange to … J: Ja, n IN 1 to, kan det både håndtere, jeg mangler nok parameteren der styrer hvor langt er jeg nået, og det skal foregå inde i BEGIN, når jeg laver en ny declare, nummer INT, (skriver og mumler). Hvad gør den nu, nu looper den over n, der går fra et, n indeholder hvor langt jeg er nået på rejsen fra et til n, som er der vi skal nå frem til, og så for hver gang så beregnet n vil være lig med, første gang vil det give nul det vil ikke du, så vi skal starte med initialisere denne her til et, hvis forudsætningen i opgaven at vi altid starter ved et. Hvordan laver vi en default værdi, den skriver vi der et, beregnet n skal jeg også have sat til et. Hvad siger programmet nu, looper fra et til n, hvor langt er vi nået i n nummer n r, den beregnet n r starter ved værdien et, første gang er den gange en med en, anden gang så gange en med to og læg det resultat, beregnet n skal have summeringsfunktionen i sig så den der plus, n som er hvor langt vi er nået, fem gange fire. F: Prøv at løbe den igennem og se, hvad er n første gang vi kommer… I: Det burde virke. F: Det gør det nemlig, med en enkelt undtagelse. Prøv at læse her, du har beregnet n, og så bruger du kolon lig med fordi? I: Jeg tildeler værdien. F: Hvordan kan man se, at det er værdien, du tildeler? I: Hvor har vi cheat sheet, er det det med at tilgå, og det samme med nr., okay. F: Så vidt jeg kan se vil det være brugbart, skal vi hoppe videre til opgave 4.	

Tid	Tale	Resultat og bemærkninger
43:55 **Opgave 4**	I: Okay, hvad siger vi så. (Læser og mumler). Så nu skal vi lave en rutine, der returnerer en tabel. Ellers så er grundskallen det samme, som tidligere. Var der noget specielt, når der skulle en tabel tilbage, tablefunction det var vel ikke det, det er mere noget med at man skal have en tabel tilbage. F: Men det er rent faktisk det, der ligger i en tablefunction. Hvis du kigger her… I: Okay, det er en funktion, som den der. Man kalder den bare et navn. F: Du skal ligge kigge her i opgaven for du får lidt hjælp. Her har du både dit schema, repository og tabelnavn. I: Okay så det er, function så bruger man dobbelt, schema navn forsvaret, kolon mellem, så pakke, kolon kolon, tabelnavn, og så har vi så en inputparameter titel, og den har VARCHAR, returns table, en inputparameter titel, det skal så forståes sådan, at der skal den kategori og den kategori ud. F: Der er et eksempel hernede. Hvis man kalder funktionen med kaptajn, så får man det output. I: Det er så sådan, det skal præsenteres. (Mumler), language, SQL Security invoker, navn, kategori, timeløn, FROM, Forsvaret, personel, WHERE, det var titel vi var ude efter, titel lig med kolon og så inputparameteren titel. Okay, hvad har vi skrevet, vi har lavet en funktion ud fra tabel vi kalder i skemaet forsvaret og i den pakke, der hedder forsvaret pac, hvor vi gerne vil tilgå personeltabellen. Der har jeg måske skrevet for meget. F: Du skal angive funktionens navn. I: Den har en inputparameter, der hedder titel, og en returtype, der er en tabel, der har tre kolonner, kategori og timeløn, og så bruger vi sproget SQL Script, så kommer SQL Security invoker as, det kan jeg ikke lige huske. F: Kan du huske det fra HANA150? Her kan man styre om funktionen skal kaldes med de rettigheder, du har haft som designer eller kaldes med brugerens rettigheder. Man vil næsten altid sige, at de skal kaldes med brugerens rettigheder.	I er usikker på hvordan funktionen navngives. I er usikker på kald med sikkerhedsrettigheder (SQL SECURITY INVOKER).

Tid	Tale	Resultat og bemærkninger
	I: Okay, så kommer ned i selve programblokken, hvor vi har retur, som også er standard, og så har man ellers SELECT statement, hvor vi siger, at vi vil hente de tre kolonner navn, kategori og timeløn, fra schemaet forsvaret og tabel personel og det skal den gøre for de rækker hvor titel er lig med inputparameter titel. F: Jeg er usikker på om den vil kaste op over at de to hedder det samme. I: Jeg kalder den input, så er det også nemmere at læse. F: Den her gang fik du kolon med. Det er fint. I: Ja, det er fordi jeg vil tilgå værdien af variablen. Det burde jo virke, det der. F: Det tror jeg også at det gør. Er der noget med parenteserne? Nej, det ser rigtigt ud. Prøv lige at læse opgave fem igennem.	
54:20 **Opgave 5**	I: (læser opgaveteksten højt) Så her skal jeg ikke returnere noget, jeg skal bare fylde tabellen, så det ligger i tabellen. Tabellen findes i forvejen, så det der er opgaven, er at lave en funktion, der med de inputparametre genererer en række tilfældige tal under hensyntagen til nedre og øvre grænser. F: Du siger, at du skal lave en funktion? I: Rutine står der. F: Prøv at vente med at begynde at skrive. Jeg tror ikke, at vi når så meget, så jeg vil hellere høre dine tanker. Hvad tænker du umiddelbart om output? I: Jeg skal lave en INSERT Table, jeg skal først have en logik, med nedre og øvre grænser, og tal skal ligge inden for spænd, der var antal rækker, så jeg skal loope antal gange, og hver gang skal jeg finde et tilfældigt tal inden for spændet, og det tal skal komme ind som en ny række i tabellen, det er egentligt det, der er opgaven. F: Så det, du siger er, at du fylder data ind i en tabel, der er der i forvejen, så du ikke har et output. I: Jeg har det output, at der kommer data ned i tabellen.	

Tid	Tale	Resultat og bemærkninger
	F: Det, der er karakteristisk for de tidligere, er at der kan du ikke skrive data, der kan du kun læse værdier ud. Og her er dit ræsonnement helt rigtigt. Du skal lave noget der fylder data ned i databasen. I: Så det er den der, jeg skal have fat i. Og den der RAND er tilfældighedsgeneratoren. F: Præcis, du kan se eksempler her. I: Og den vil så tage hensyn til grænserne. F: Nej, det vil den ikke. Der skal man selv lave matematikken. Det der var væsentligt for mig, var det med INSERT INTO. I: At der ikke er en returparameter. Men at det ryger ned i system. Det er egentligt meget spændende.	
58:32 Afslutning	F: Du skal ikke skrive mere. Jeg skal bare høre dig, om der var noget, der var svært eller noget der var nemt, om der var noget hvor du tænkte, at det skal man lige vænne sig til? I: Det gode er jo faktisk Wordpad, man får ikke nogen hjælp, så man bliver tvunget til at tænke over, hvad man skriver. Det er både godt og skidt. Syntaks er noget gange det, der kan stoppe en meget. Hvis man sidder og har de overordnede tanker, så kan en syntaksfejl blokere en, der har man jo som regel nogle gode editorer, som kan hjælpe en på vej. Det gode her er, at man prøver forskellige funktioner, en med en parameter, en med to parametre, en med returtabel, og så i opgave 5 med at prøve at skrive noget ned – en hel anden type returresultat, der enden nede i databasen. Det er meget funktioner og ikke noget med at formatere, men det dækker sikkert dit formål med cheat sheet, på den måde er det meget tilgængeligt. F: Der var kolon foran variabelnavne, der drillede dig et par gange. I: Ja, det ville jeg nok få at vide (af editor/compiler). Hvert sprog har sin egen syntaks, hvordan slutter man en linje, hvor følsom er den over for store og små bogstaver. Det der kolon foran, er noget man skal være opmærksom på.	

Tid	Tale	Resultat og bemærkninger
	F: Jeg var skide irriteret over det de første timer, da jeg rodede med det. Så var der lidt med at definere DECIMAL. I: Jeg havde ikke læst det grundigt nok..., at det var længden. F: Så talte vi om det med at bruge ELSEIF hele vejen ned. I: Og her, hvor man har en der er hel åben til sidst, så vil de to første IF fange dem, man kender og alt andet ryger ned i slutbunken. F: Hvad siger du her, hvor man først arbejder med noget FOR og IF-THEN, og så er man pludseligt ovre i noget SQL, det med at blande, tænkte du over det undervejs. I: Ikke synderligt, det er selvfølgeligt noget andet, hvor man skal ned og hente resultatet, der findes i forvejen, mens det andet er noget, hvor man beregner et resultat. Jeg skelnede ikke lige, at der var forskel på den måde i opgaverne, men det kan jeg god se, nu hvor jeg får det at vide. F: Hvis vi var kommet videre i opgave 5 ville du have oplevet endnu mere sammenblanding. Hvis du har arbejdet i ABAP er du vandt til blanding af begge dele. I: Ja. F: Hvornår så du SQL første gang. I: Det gjorde jeg i år 2000. Da jeg var færdig med Cand. merc. startede jeg som web programmør. Det var på SQL server. F: Så har du set det lige siden. I: Ja, men det var i en anden kontekst end SAP. F: Ja, men som du selv siger, har hvert sprog sin egen variant. Imperativ programmering med loops og forgreninger har du også arbejdet med. Så man kan godt sige, at du har lang erfaring med begge dele. I: Ja, jeg blev undervist i C++ på Cand. merc., med klasser og objekter. Det blev desværre kun til studietiden.	

Resulterende kode:

OPGAVE 1:

```
CREATE FUNCTION CONVERT_PUND_TO_KG (PUND DECIMAL(8,4)) RETURNS KG (8,4) AS
BEGIN
KG := PUND/2.2046;
END
```

OPGAVE 2:

```
CREATE FUNCTION BEREGN_TIMELØN (KATEGORI VARCHAR(30), TIMELON INTEGER)
RETURNS  BEREGNET_TIMELON DECIMAL (7,2) AS
BEGIN
  IF :KATEGORI = 'MILITÆR' THEN
    BEREGNET_TIMELON := TIMELON * 1.4;
  ELSEIF :KATEGORI = 'CIVIL' THEN'
    BEREGNET_TIMELON := TIMELON * 1.6;
  ELSE
    BEREGNET_TIMELON := TIMELON;
  END IF;
END
```

OPGAVE 3:

```
CREATE FUNCTION OPLOFT_N (N INT)
RETURNS BEREGNET_N BIGINT AS
BEGIN
 DECLARE NR INT = 1;
 BEREGNET_N := 1;
 FOR NR IN 1 .. N DO
   BEREGNET_N := :BEREGNET_N  * :NR;
 END FOR;
END
```

OPGAVE 4:

```
FUNCTION "FORSVARET"."FORSVARET_PAC::tfPersonel"
  (in_TITEL VARCHAR(30)
```

```
RETURNS TABLE (NAVN VARCHAR(3),
                              KATEGORI VARCHAR(30),
                              TIMELON INT))
LANGUAGE SQLSCRIPT
SQL SECURITY INVOKER AS
BEGIN
  RETURN
   SELECT NAVN, KATEGORI, TIMELON
   FROM "FORSVARET"."PERSONEL"
   WHERE TITEL = :in_TITEL;
END;
```

Transskribering af forsøg med IBM medarbejder nummer 4.

Fil: 2019-10-01_11-33-48.mp4

Gennemført 2019-10-01.

Tid	Tale	Resultat og bemærkninger
2:16 Indledning	F: Om lidt så stikker jeg dig et cheat sheet, som du får mulighed for lige at læse igennem, så sætter jeg mig hen og laver noget andet i fem minutter. Du skal ikke gå ind i koden i alle eksemplerne linje for linje, men finde ud af hvad overskrifterne er, og se hvad det er for nogle eksempler, der er for at få et overblik. Sproget vi bruger er SAP HANA SQL Scripting, det ved jeg ikke om du kender? I: Nej. F: Det er det nye, når man vil programmere op imod en HANA database, hvis man kender SQL i forvejen eller ABAP i forvejen, så er det ikke alt for fremmed, det kan vi komme tilbage til. Når du så har brugt fem minutter på at finde ud af, hvad der er op og ned på det, så får du et task sheet. Der er fem opgaver, så du så stille og roligt går i gang med at løse. Du har det mest lækre udviklingsværktøj i dag – nemlig Wordpad, så du får ingen hjælp ud over cheat sheet og så mig, for som jeg også sagde til introduktionen, så går det ikke ud på at teste jer, det er et spørgsmål om at teste den model ved at tage nogle ind og give dem en times tid, og se hvad jeg kan uddrage af data på den baggrund. Derfor bad jeg jer også om at tænke højt. Det har de andre faktisk været rigtigt gode til, og det hjælper lidt, når vi sidder her ved siden af hinanden, og jeg spørger dig lidt og peger på skærmen og spørger, hvad har du gang i her, eller du kører fast i et eller andet og siger, jeg ved ikke lige hvad jeg skal her, og så siger jeg, at du måske skulle kigge på den anden side af cheat sheet under punkt et eller andet, eller du siger, her kunne jeg faktisk gøre det en måde, men også på en anden måde. Det er alt sammen med til at give mig nogle data, som jeg kan tage med hjem og arbejde videre på. Når vi så er færdige, så tager vi lige en hurtig snak om hvad du synes var nemt, var der noget der drillede undervejs, og så lige til sidt kort om din baggrund, lige det her noget, du har set før. Det her sprog kan du relatere det til noget du kender som fx SQL eller ABAP, måske du siger at her er noget der lidt anderledes end det, jeg er vant til. Bortset fra når du lige får lov at hygge dig i fem minutter, så sidder vi her ved siden af hinanden og har snakken stille og roligt. Med lidt held, så	I virker afslappet og naturlig under forsøget. I er spørgelysten og ivrig.

Tid	Tale	Resultat og bemærkninger
	er du også blevet lidt klogere. De andre har i hvert faldt sagt, at det har været givtigt, så det håber jeg også for dig. Er du frisk? I: Det kan du tro? F: Så får du lov til at starte med cheat sheet.	
14:55	F: Er vi ved at være der? I: Ja, det er hårdt, for en der kun kan sådan noget BW programmering. F: Det gør ikke noget. Nu skal du se her, det er opgaven, her øverst er der en tabel, der skal bruges i de senere opgaver, den er ikke relevant for opgave 1, så jeg vil foreslå, at vi går tilbage til den, når vi støder på den, og så i stedet gå i gang med opgave 1. I: (læser) Man skal bruge nogle variable. Vi skal vel i princippet bruge en inputvariabel og en outputvariabel, og den havde vi heroppe, (skriver) vi kalder den input pund, så kan vi huske det. F: Jeg kunne godt forestille mig, at jeg kom med en dims der vejer tre et halvt pund. I: Ja, lige netop, en varchar, det er kun karakterer, decimal kan den forkortes, DEC, så skal man have en parentes med, ti komma to, det må jeg regne igennem, output der skal vi have kilo, også decimal fjorten komma fire, det ville jeg justere senere, jeg er lidt i tvivl om hvordan jeg skal slutte. F: Jeg har skrevet en rutine i opgaven for at lade det være op til dig om du vil lave en funktion eller en procedure eller hvad du vil. Jeg tror, at du skal prøve at vende cheat sheet om, så kan du få lidt hjælp til at komme i gang. I: Nå ja, den sender én retur. Det står jo fint, det er jo stort set den, det er tæt på. F: Men du havde et fint ræsonnement om, hvad du skal have af input og outputvariable. Men jeg savner nok noget i rutinen, der regner lidt. I: Ja for katten, det er fordi jeg ikke er færdig, det er helt korrekt, der skal stå mine pund, dem skal jeg dividere med 2 komma 2046. Den var kommet i testen.	I resonerer godt vedr. valg af variable, både input og output. I bemærker, at han nok vil gå tilbage til antallet af decimaler i variablerne. I er så optaget af cheatsheet, at han glemmer logikken i rutinen.

Tid	Tale	Resultat og bemærkninger
	F: Og så lige semikolon efter teksten. I: Men det står der ikke i cheat sheet. F: Nej, det er rigtigt, og det går fint, når der kun er en funktion, men hvis der havde været flere efter hinanden. Så begge dele kan være rigtigt. I: Jeg gik mere op i syntaksen her. F: Det var opgave 1. Du må meget gerne gå videre til opgave 2.	
27:39 **Opgave 2**	I: (Læser og mumler) Det er til at forstå det her. Vi har inputparametre står der her. Vi skal ikke læse tabellen der. Så det er ligesom den heroppe. Det er det BW drejer sig om (cut and paste). Så den skal hedde, vi kalder den kalk_overtid, der har vi så to værdier. Der har vi to værdier, kategori og timeløn, så kan jeg næsten gætte, at det ser sådan her ud, nej måske mellemrum, adskillelse. Nu tager vi lige hovedtrækkene, så det fine bagefter. Vi laver lige skabelonen, det var vores input, så laver vi output. Vi kalder den overarbejde, sådan, så kommer beregningen dernede, vi laver en begin, så er jeg sikker på en IF, hvor har vi en IF i cheat sheet, det er jo bare den her skabelon, vi kører løs på. Hvorfor siger du ikke det? F: Jeg er mest interesseret i, at høre dine ræsonnementer og se processen. I: Så jeg skal have et komma deroppe. Okay, timerne, 14 er alt for meget. F: Nu siger du timer. Du har kaldt den kategori. I: Ja ja, det er en varchar og integer, der er ikke nogen længde på. F: Nej, den er altid 32 bit lang. I: Ja, jeg kører lige over på den der igen. Det var den der, der kommer mit resultat af overarbejde, der må jeg skulle gange med en komma fire, en komma seks eller ingen ting. Teoretisk set skulle jeg (uforståeligt). Der kan stå et par tusind timer, jeg har ikke noget imod at have rigeligt. Det er jo ikke sådan at man dør på diskplads i dag, det var i gamle dage.	I virker lidt usikker på definitionen af en DECIMAL, i hvert fald som output til en timeløn. I resonerer fint i forhold til brug af IF-ELSIF-ENDIF. I er usikker i brugen af kolon til at referere en variabel. I er usikker på forskellen mellem kolon og kolon lig med.

Tid	Tale	Resultat og bemærkninger
	IF, nu skal jeg test på inputkat, hvis den er lig med militær, det skal vi nok angive på en eller anden sjov måde. Vi kontrollerer, det kan jeg godt lide, hvis han er militærmand så, skal vores resultat, det er den der, (skriver og mumler), det er en INTEGER, vi får fat i der, så skal vi gange den med en punktum fire, sådan der. Så har vi taget den på militær, nu skal jeg lige se logikken, for den slutter på, at hvis vi ikke rammer nogen af de to første, så tager den bare de sidste. Jeg kunne også lave en ny IF, og så sige jeg kunne tage en tredje og så sige, at hvis der var noget andet, så ville det slet ikke komme i, men det gør vi ikke. Vi følger denne her. Hvad hed de, civil gange en komma seks, der mangler noget her, det ville den tage, når jeg tester, der er den SAP test knap, så skal den tage det samme men bare gange med en komma seks, og så slutte med ELSE, den skal ikke have noget, gad vide om den skal have et hak i enden, det skulle den, og så en END. Så skal jeg lige løbe det igennem. (Læser og mumler). Nå hvad siger vi, jeg kan ikke teste. F: Så kan jeg være compileren. Jeg har et par spørgsmål. Det der kolon, kan du fortælle mig lidt om det? I: Den skulle gerne hente oppe fra variablen deroppe. F: Du bruger kolon for at få værdien. I: Ja, fra variablen. F: Kan du så sige noget om den der variabel, hvad er det du gør? I: Der henter jeg den anden variabel, vi lægger ind. Jeg får to variable ind om det er militær eller civil og en timeløn. Så tester jeg om det er en militær fyr, det var det, så siger jeg okay, så skal jeg beregne overarbejde, men i input heroppe har de måske lagt 300 kr. ind, så tager jeg… F: Hvordan indikerer du, at det er indholdet af variablen du bruger? I: Det gør jeg med kolon. F: Det var intentionen. I: Det (kolon) er på den forkerte side. Det kan jeg godt forstå forvirrer. F: Jeg bliver ikke nødvendigvis forvirret.	

Tid	Tale	Resultat og bemærkninger
	I: Det var sådan her, det var tiltænkt. Output lig med, så tager jeg det, der er i variablen, og sætter overarbejde lig med det der i variablen gange en komme seks. For at test om du var vågen. F: Prøv at kigge i eksemplet. I: Nå for pokker, der er jo mange. F: Det er derfor jeg siger, at jeg ikke nødvendigvis bliver forvirret. I: Hvorfor gør man det sådan. Hvordan skal man tolke det. Dernede skal der være kolon på output. F: Kolon lig med fortæller, at nu tildeler du variablen en værdi. Kolon foran en variabel, betyder, at nu tager du værdien i variablen. Så heroppe lægger du ikke noget nyt i variablen, du laver blot en test på den, men hernede lægger du noget nyt ind. I: Så kan jeg ikke helt se forskellen på den her og den der oppe. F: Her siger du, det er variablen du har fat i, som du vil have indholdet af. Heroppe er det bare en streng. Det var en af de ting, som jeg selv fandt bøvlet i det her sprog, og som mine analyser viste nok skulle komme til at drille nogen. Det har du nu bevist, det var fint. Vi hopper videre til opgave 3.	
46:50 **Opgave 3**	I: (læser og mumler) Nå, en input og en output, så vi kan jo i princippet bruge den igen fra opgave 1 som en lille skabelon, og siger vi at vi beregner x! er det noget vi selv har fundet på til opgave. F: Det er en matematisk funktion, den hedder fakultet. Hvis jeg beder dig om 3 fakultet, så er det tre gang to gange en osv. I: Så vi skal have den vi sætter i fakultet. F: Du får lidt hjælp her, for den siger, at der er en inputparameter. I: Det er x ja, vi kalder den for fakultet, input x, det er et heltal, det er bare INT. Output, hvad kan det være, result. Og	I har forståelse for variabel/objecttyper. I starter med at skitsere loop med papir og blyant. I er usikker i reference til variable, herunder brugen af kolon. I bruger ikke semikolon til afslutning af udtryk korrekt. I anvender ikke kolon lig med ved tildeling af en værdi til en variabel.

Tid	Tale	Resultat og bemærkninger
	den skal være stor. Der havde vi nogle specielle store typer herovre. F: Det er rigtigt, at det er øverst i midterste kolonnne, det er bare omme på den anden side. I: Den var der. F: Jeg kunne se på dig, at du vidste lige præcis, hvor du skulle pege hen. I: Det er ikke large objectsize, det er til tekster, billeder, skidt og kanel. Det har ikke noget med det at gøre. Det skal være en 64 bit, en BIGINT. Output result, der smider vi det ud, det angiver med kolon, og så henter vi input x, det var vores variable værdi, den henter vi, og løfter vi den op i det der fakultet, kan man det direkte her? F: Nej, den funktion findes ikke, så den må du selv… I: Nåh, så er jeg nødt til at lave en loop-ting på en eller anden måde. F: Det er rigtigt. I: Nåh, det var ikke pænt, hvordan tror jeg skal nå det inden for en time. Jeg skal have en loop, n er input, så kan jeg bare vælge. Nej nu skal du se, vi laver to variable, skal jeg gøre det oppe i toppen eller herinde. Jeg skal bruge dem til at regne på. F: Så skal du gøre det efter begin. Du vil have sådan en lille lokalvariabel, du kan bruge til et eller andet… I: Ja, jeg vil loope, lad os bare skrive LOOP og END LOOP. Når vi går ind her, så ved jeg, lad os bare sige, at der står fem. Vi tager lige loopet bagefter, eller skal jeg finde ud af det først (leder i cheat sheet) der, forståer jeg lige det. F: Der er to måder at loope på: enten ved man hvor mange gange, man vil loope, eller også looper man indtil et eller andet bliver opfyldt. Det er den metode og den metode. I: Jeg skal bruge en WHILE. WHILE, må jeg ændre på værdien input x, det kan være lige meget, hvis jeg ikke skal bruge den igen – ikke. While input_x, her skal der kolon på, min variabel input_x er større end nul DO, hvis input_x er større	

Tid	Tale	Resultat og bemærkninger
	end nul, det første jeg skal gøre, er at jeg skal tage, jeg starte med at begynde at beregne mit output_result, den prøver jeg lige at beregne, jeg sætter lige kolon på bagefter. Hvis jeg nu siger at der står fem, når jeg går ind i den, så går jeg ned og siger input_x den er større end nul, når der står fem, jeg skal regne videre på den, der skal jeg bare sige, at den er lig med input_x, nu skal du bare se her, så tager vi lige logikken på om lidt. Nu har jeg smidt fem ind i den, og så siger jeg OK, så tager jeg input_x lig med sig selv minus en. F: Ja, så decrementerer du den. I: (kopierer) så skal jeg have nogle koloner på bagefter, så tager vi lige og kigger på det. Der står fem i input_x, så det er større en nul, nede i output_result der står nul, så tager jeg, det dur ikke, for der står nul, så den vil give nul gange fem, det giver ingenting, så vi skal lige have en IF på. Vi siger IF output_result er større end nul, så skal den tage det der og gange med hinanden, og ELSE så skal den tage input_x, det er den første loop. Nu siger vi, at der står fem derinde, output_result er ikke større end nul, fordi det er første gang vi rammer, så vi kommer herned, så står der, at output_result skal være lig med fem, så har jeg femtallet. Jeg skal lige have en ENDIF, og den der skal hedde WHILE. Vi kører WHILEn, der står fem, den er ikke større, vi går ned i ELSE, den bliver til fem, så tager vi input_x og sætter til input_x minus en, så der fire i input_x, den kører op igen – ikke. Nu står der fire, så siger jeg output_result, er den større end nul, ja der står nemlig fem OK, så går jeg herned, så siger jeg output_result er nu lig med, det dur så ikke, jo det går de da, der står nemlig fem fra før, nu er jeg ved at lægge en ny værdi i den, men der står fem fra før gange de fire dernede, den har jeg jo sat til fire nu, de bliver fem gang fire og den springer den der over, det var ELSE, så går den ned og sætter den til tre, så kører det. Det burde køre, kan du følge mig? F: Ja, jeg kan sagtens følge dig. Vil du sprede lidt koloner ud i det der? I: Ja, så laver jeg en test, og så lyser den rødt. (læser koden for at sætte koloner). Så mangler jeg måske nogle semikoloner. Jeg er lidt i tvivl. F: Det går lidt galt med kolonerne. Det kolon, det kolon og det kolon, de skal ikke være der. Der er det variablen du tildeler en værdi, det er jo ikke indholdet af variablen, du vil have fat i her, der er det selve variablen. Herhenne, der vil	

Tid	Tale	Resultat og bemærkninger
	du have fat i indholdet, men når du så har lavet den beregning, så vil du have den ind i variablen. Til gengæld når du laver tildeling, så var, at vil skal have kolon lig med. I: Hov ja, jeg skal have kolon derude, for det var der jeg skulle have den, og der, og i princippet også der. F: Ikke kun i princippet, det skal være der. I: Det er rigtigt, det var ikke derude, men derinde. F: Så tror jeg, at vi nåede igennem opgave 3. Vi kan godt nå at tage hul på opgave fire.	
1:05:30 **Opgave 4**	I: (læser) Det vil sige, at vi skal læse den tabel deroppe. Det bliver lidt spændende, nu skal vi til at lave noget nyt. Vi skal læse en tabel, og vi skal skrive en tabel. F: Dit output, hvad er det? I: Det er lidt uklart. Det er en tabel, men om den er fysisk, eller hvad. Jeg vil skyde på, at det er en intern tabel til at starte med. Nåh, vi skal have lavet en tabel. Vi må gå ud fra at den tabel er der. Så. F: Du skal skrive en rutine, der har en parameter, rutinen skal returnere en tabel. Du skal have fundet en rutine, der spytter en tabel ud. Indtil nu har du spyttet værdier ud. I: Jeg skal lave en create function af en art. Det er kun et spørgsmål om, at outputtet er en anden type – en table. F: Det er i hvert fald rigtigt, det er et andet output. I: Det er jeg lidt i tvivl om. F: Du skal nok se på den anden side. Det er de her, vi har brugt indtil nu. De har den egenskab, at de spytter en værdi ud. Herovre er der noget der hedder user defined table function. Prøv at se, hvad den returnerer. I: Returns table, ahh. Ja, det er jo den der. Det kan jeg godt få øje på. Nå den hedder function, titel, (læser), schema er det nede på SQL tabellen? F: Den tabel du får opgivet, den ligger i et schema – se her. Der er et schema, der hedder forsvaret, og der er et repository, der hedder forsvaret underscore PAC.	F støtter I for at komme i gang med opgaven. I er usikker på brugen af schema og repository. I er usikker på hvad det vil sige at returnere en table. I efterlyser en navnestandard. Retter efterfølgende til mere sigende navne for variable og funktion. I virker mere sikker i SQL delen af opgaven – sammenlignet med SAP HANA SQL Scripting i de foregående opgaver. Erkender forskellen på variabelnavne og kolonnenavne.

Tid	Tale	Resultat og bemærkninger
	Så vi har schema, hvor vi har tabeller liggende og så repository, hvor vi lægger koden nede i HANA databasen, ligesom her har vi også fat i et schema og en package. Tilsvarende skal du have fat i forsvaret og forsvaret underscore_pac. I: Det kan jeg se (skriver og mumler). Hmm, dobbelt kolon? F: Jeg kan godt forstå, at du spørger, den er ikke nem, slet to tegn og så indsæt to koloner. Nu har du først angivet, at det ligger nede i den database, der hedder forsvaret og den kode, du laver nu, skal ligge i forsvaret underscore pac, så nu skal vi bare have den navngivet, så den skal finde et godt navn til den funktion, kolon kolon og så kalder du den funktion – titel, og så skal du afslutte. Så kalder du med en eller anden parameter. Nu har du defineret navnet på din funktion, og sagt til databasen, hvor den skal lægge den henne, så skal du have den parameter, du kalder funktionen med, her er det en kategori. I vores opgave er det titel. Så nu har vi styr på funktionen. I: Så skal vi have en return table, og den har jeg derovre, den har jeg lige givet, nu bliver jeg lidt forvirret. Var det ikke min tabel, jeg har placeret her? F: Her fortæller du, at du vil ned i schemaet forsvaret i repository pac, og der vil du have oprettet en funktion, der hedder titel, når man kalder den funktion, skal den have en inputparameter, en streng, og når den returnerer noget, skal den returnere en tabel, og nu skal du definere den tabel. I: Så skal jeg skrive, at den hedder titeltab, så kan vi hidde rede i det, jeg har min funktion her nede og min table heroppe. Her skal jeg bare skrive indholdet i den, den kommer til at hedde titel, navn (skriver). Sådan der, så skulle den være afsluttet. Language, det er bare noget man skal vide. F: Language, så fortæller vi databasen, at det der kommer er SQL scripting. Og her fortæller vi den, at når den kører det der, så skal det køres med de rettigheder brugeren har, altså den der kalder funktionen, den persons rettigheder skal det køres med. I: Det vil jeg have ned i tabellen, så jeg kan bare selecte det jeg kender fra tabellen derovre. Hvorfor laver vi en RETURN her?	

Tid	Tale	Resultat og bemærkninger
	F: Det er for at fortælle, at det er tabellen, der er output og ikke bare lave SQL kaldet. I: Det var den funktion, jeg søgte, så det er godt nok. SELECT navn, kategori og timeløn. (Skriver og mumler). FROM schemaet forsvaret – ikke? F: Lige præcis. I: Jeg skal ikke angive noget repository der. Tabellen ligger i schemaet, så det er egentligt bare navnet. WHERE titel er lig med den titel, jeg havde deroppe. Jeg tror jeg ville finde en eller anden måde til at styre navne, hvis jeg skulle gøre det i virkelighedens verden. F: Kan du huske at du spurgte om, hvorfor de starter med at kalde dem e x under score og i m underscore variablenavnene. Det er for at indikerer at det er output og input variable. Så der findes en metodik. Du har gjort det lidt svært for dig selv, for du bruge titel til tre forskellige ting lige nu. I: Hvis jeg kalder den for min input, min inputvariable (retter til). (Drøftelse af navngivning og tilretning). Nu giver det mening. Den var værre den her. (Gennemgår koden og mumler). Det tror jeg var det. F: Det var det.	
1:26:26 Afslutning	F: Nå, hvad var nemt og hvad var svært? Fik du varmen. I: Det er lang tid siden, at jeg skulle skrive så meget. Når jeg laver noget, så er det tit jeg finder noget, der ligner. Vi laver typisk opslag på noget, vi har gjort hundrede gange før. F: Det gør alle nok. I: Indimellem når vi skal lave noget stort, så tager vi ham, der er bedst til det. Eller vi finder en ABAPer, til det der ikke ligger til højrebenet. F: Jeg har skrevet, at du gør brug af genbrug. Det er der ikke noget forkert i. I: Problemet kan være, at når fejlen er lavet en gang, så kommer den igen.	

Tid	Tale	Resultat og bemærkninger
	F: Jeg har skrevet at du i hver opgave ræsonnerer hvad der er input og output. Så har jeg skrevet, at de koloner driller dig. Det med at få fat på indholdet i variablen og kolon lig med, når man tildeler noget til en variabel. I: Jeg tror det skal prøves nogle gange, så sidder det der. Jeg savner min test, når jeg gør det normalt, så kører jeg små bidder og tester hele tiden. Hvis jeg bare skriver et bjerg, så kan jeg ikke se hvor den stopper, jeg kan ikke se hvor fejlen er, fordi der er for mange af dem. F: Det er rart, når man er i et miljø, hvor man kan. Der er jo ikke altid, man kan teste om tingene virker. Det kan man i SAP. Hvis man nu fx skriver kode til sådan en fætter her (Arduino), så må man oploade det i den, og så se om det virker når man er færdig. Enten så virker det, eller også virker det ikke. I: Jeg har siddet på KBH Uni og oploadet kode til en VAC, det var det sammen. Det er mange år siden. F: Så talte vi lidt om at IF, ELSEIF, ELSE, ENDIF kunne bruges på forskellige måder. I: Ja og WHILE. F: Ja, du kunne også have brugt FOR-NEXT til at loope. Du gjorde måske en lille smule tungt, for fem gange fire gange tre gange to gange en er det samme som en gange to gange tre gange fire gange fem, så du kunne bare… I: Det er rigtigt, jeg kunne bare have talt opad. Som en FOR. Men det var nok fordi jeg var begyndt at lave loop herovre (på et stykke papir ved siden af tastaturet), hvor jeg havde defineret min egen tæller, den var jeg i gang med, så derfor kørte jeg ud af den streg. F: Du er faktisk den eneste, der har siddet overhovedet og noteret noget ved siden af, og det er noget af det man analyserer i nogen af de andre analysemodeller om vælger på et tidspunkt at bruge andre hjælpemidler, end lige computeren og skærmen, det var faktisk ret interessant at du gjorde det. I: Jeg skulle have hovedet væk fra det. F: Du ser også at tit, at to programmører straks går i gang ved tavlen.	

Tid	Tale	Resultat og bemærkninger
	Du har skrevet SQL før. I: Ja, SQL og ABAP. F: Du har også kendskab til imperativ programmering. Det har du, hvis du har skrevet i ABAP. Du har været vandt til at bruge fx IF og THEN. I: Ja ja. Og FOX koden, som vi bruger i BPC.	

Resulterende kode:

Opgave 1

```
DECLARE INPUT_LBS DEC(14,4)
DECLARE OUTPUT_KG DEC(14,4)

CREATE FUNCTION convert_LBS (INPUT_LBS DEC(14,4))
RETURNS OUTPUT_KG DEC(14,4) AS
BEGIN
  OUTPUT_KG := :INPUT_LBS / 2.2046;
END;
```

Opgave 2

```
CREATE FUNCTION KALK_OVERTID (INPUT_KAT VARCHAR(30), INPUT_TIMELOEN INT)
RETURNS OUTPUT_OVERARB DEC(10,2) AS
BEGIN
  IF  :INPUT_KAT = 'Militær' THEN
    OUTPUT_OVERARB := :INPUT_TIMELOEN * 1.4;
ELSEIF   :INPUT_KAT = 'Civil' THEN
    OUTPUT_OVERARB := :INPUT_TIMELOEN * 1.6;
ELSE
    OUTPUT_OVERARB := :INPUT_TIMELOEN;
END IF;
```

```
END;
```

Opgave 3

```
CREATE FUNCTION FAKULTET (INPUT_X INT)
RETURNS OUTPUT_RESULT BIGINT AS
BEGIN
WHILE :INPUT_X > 0 DO
IF :OUTPUT_RESULT > 0
  OUTPUT_RESULT :=  :OUTPUT_RESULT * :INPUT_X
ELSE
  OUTPUT_RESULT :=  :INPUT_X
END IF;
  INPUT_X := :INPUT_X - 1
END WHILE;
  END;
```

Opgave 4

```
FUNCTION "FORSVARET"."FORSVARET_PAC::LOKALLOEN"

(Titel_INPUT VARCHAR(30))
RETURNS TABLE (Navn VARCHAR(30), Kategori VARCHAR(30), Timeloen INT)
LANGUAGE SQLSCRIPT
SQL SECURITY INVOKER AS
BEGIN
  RETURN
  SELECT Navn, Kategori, Timeloen
  FROM "FORSVARET"."Personel"
WHERE Titel = :Titel_INPUT ;
END;
```

Transskribering af forsøg med IBM medarbejder nummer 5.

Fil: 2019-10-03_10-40-58.mp4

Gennemført 2019-10-03.

Tid	Tale	Resultat og bemærkninger
03:43 Indledning	F: Vi går i gang om lidt. Og så tager det max 90 minutter, det har taget en times tid med de andre, og så er der lidt snak før og efter. Det det går ud på, er at lige om lidt får du det her cheat sheet. Det sprog vi hygger os med i dag er HANA SQL Scripting, så hvis du har læst det materiale, som blev rundsendt på et tidspunkt HANA150, så vil der være noget du kan genkende. I: Det har jeg ikke. F: Men det kommer vi til at leg lidt med. Du får fem minutter, hvor jeg laver noget andet, hvor du kan læse det her. Du skal ikke sætte dig ind i alle detaljer, men prøv at få et overblik over de tre kolonner på begge sider af arket, så du har en idé om, hvor du kan finde tingene henne. Når du har brugt tid på at lære det at kende, så får du opgaverne her, der er fem opgaver, og jeg har ingen forventning om, at du når dem alle sammen, og det er heller ikke vigtigt. Det værste, der kan ske er, at du sætter dig ned og løser alle opgaverne uden at lave fejl, for så får jeg ikke data. Det foregår ved, at jeg har givet dig det bedste udviklingsværktøj der fås, nemlig Wordpad, og det er det, du kommer til at skrive i. Opgaverne har stigende sværhedsgrad, så i starten vil du mere eller mindre kunne finde en skabelon i cheat sheetet, og sige det er her jeg finder skelettet og fylde mit eget content i. Så stiger sværhedsgraden undervejs, og du kommer rundt i forskellige områder. Jeg kommer til at sidde her undervejs, og hvis du kører fast i et eller andet, siger du det blot, så siger jeg måske, det er fint prøv at kigge der i cheat sheet, det kan også være jeg siger, at det er irrelevant i forhold til det, vi er her for i dag, så hop blot videre. Det kan også godt være, at jeg på et tidspunkt peger på skærmen og siger, nu har du skrevet sådan, hvad har du tænkt på, eller hvorfor, så det er en dialog, og jo mere du siger, jo bedre er det. Husk, at det er ikke dig, der er under test, det er modellen, og det er et spørgsmål om at få data i kassen. Det du laver på skærmen bliver optaget, jeg har slået Snagit til, og når vi er færdige gemmer jeg det. Til sidst tager vi en snak om, hvad der var nemt og svært,	

Tid	Tale	Resultat og bemærkninger
	noget du gentagne gange ledte efter. Til allersidst taler vi om, om det ligner noget du har set før. Det er jo en SAP verden, så det kan være at du siger, at det ligner meget ABAP eller SQL. Du får denne her.	
14:45 **Opgave 1**	F: Nu skal du se, i opgaven her er der først en tabel, som bliver brugt i nogen af de senere opgaver, men den er faktisk ikke nødvendig for at løse den første opgave, så jeg vil faktisk anbefale dig at gå lige til opgave 1, så kan vi vende tilbage til tabellen. Så opgave 1 er herfra og derned til. Kig på den. I: Okay, skal jeg bare tænke eller tale, jeg tænker at det ligner meget denne her funktion, så det er det, der skal til. Så jeg vil bare skrive det her, os så lave det om undervejs. F: Du har to eksempler, et med en inputparameter, og et med to inputparametre. I: Så kan jeg nøjes med denne. F: Prøv at fange an, du får ikke meget hjælp af udviklingsværktøjet. I: (Skriver) Somme tider kan man ikke regne ud, om det har signifikans om den hedder sådan (variabelnavn starter med im_), men det tror jeg ikke. F: Nej, det er en SAP notifikationsform. I: (Skriver) Det vil være mit gæt. Det vil jeg tro, er tæt på. Med fire decimaler. F: Et par spørgsmål. Den her im_pound, det er din input parameter for pound. I: Ja nåh, jeg har skrevet hours heroppe. (retter). F: Hvad er det jeg kommer med i pund? Jeg kan også spørge på en anden måde. Hvis vi finder en håndgranat ude på lageret, så vejen den et komma to pund. Vil din funktion kunne håndtere det? Det jeg spørger ind til, er at din inputparameter er integer. I: Ja, det skal vel også være decimal. F: Hvis du kigger i cheat sheet, så er det rigtigt at DECIMAL	I er usikker på definitionen af DECIMAL(x,y). Glemmer semikolon efter END.

Tid	Tale	Resultat og bemærkninger
	tager to parametre, du har skrevet fem komma fire, har du en fornemmelse af, hvad de to tal betyder? I: Det kan jeg se her, der står total og fraction. Så det kan højest være et et kilo eller et pund, der skal lidt mere med. Vi siger 10. F: Så har du ti cifre i alt, hvoraf der kan være fire cifre på højre side af decimalkommaet. Det er fint, lad os hoppe videre til opgave 2.	
23:00 **Opgave 2**	I: Det har også lidt at gøre med hvilken stil man lægger for dagen. Nogle gange skriver man noget, og ser hvad compileren siger, eller man kører det og ser, at der kun er to cifre foran kommaet. F: Enig, og ofte starter man jo ikke på et helt blankt stykke, man finder noget man har lavet før. I: Ja, så finder man noget fra 2012, og så er det ikke lige det bedste. (læser) Ja, så må det jo være den lidt mere avancerede. F: Hvad får dig til at sige det? I: Vi skal have to parametre. Jeg vil starte med at kopiere denne her oppe. (Kopierer og skriver). F: Du kan få lidt hjælp heromme, her kan du se hvordan strengen er defineret. Så det er en VARCHAR på 30 karakterer. I: (Skriver) Det er nok kroner, men resultatet det er ikke nogen INTEGER. IF, mon den kan tage æ´er. F: Det kan den i hvert fald inde i en streng. I: THEN, lige nu er der kun de to muligheder, så alle andre end militær eller civil får ingen sats. F: Der står her, at hvis kategorien hverken er militær eller civil, skal outputtet være timelønnen uændret. I: Gennemgår koden (og mumler). Nu er jeg egentligt godt tilfreds. F: Så har jeg et par spørgsmål. Den der skråstreg, hvad betyder den?	I kommer til at bruge division i stedet for gange. Fejlen vurderes ikke at være afhængig af sproget. I er usikker på brugen af kolon foran variable, men resonerer sig frem til betydningen. I er usikker på forskellen på lig med og kolon lig med. I mener at lig med i begge situationer er nok idet konteksten forklarer resten. Glemmer semikolon efter end.

Tid	Tale	Resultat og bemærkninger
	I: Den betyder at man ganger (Griner). F: Du bliver ikke gode venner med dem, der skal have ekstra overarbejdsbetaling. Her har du et kolon foran den variabel, har du en fornemmelse af hvad det kolon betyder. I: Det betyder, at jeg referer til variablen kategori. F: Ja, og hvad betyder det, når du refererer til den? Hernede har du en anden variabel, nemlig din output variabel, der har du ikke kolon foran. I: Nej, det er rigtigt. F: Hvad tænker du om det? Du skal ikke rette det. I: Det er ikke noget man nødvendigvis ser i andre programmeringssprog. F: Det er jeg helt enig med dig i. I: Her erklærer vi at den er lig med det der, her henviser vi til variablens. F: Det er rigtigt, med kolon foran siger vi tag værdien af variablen. Det er rigtigt som du siger, her erklærer vi, hvad den der skal være lig med, og herovre vil vi bare have værdien. I: Her er der også kolon foran lighedstegnet. F: Lige præcis, det er der heroppe, men det er der ikke der. Det er meget godt set. Hvad tænker du om det? I: Det er vel for at tydeliggøre om det er en betingelse. F: Ja, når det står som kolon lig med, så er det en tildeling. Her tildeler vi den værdi herovre til variablen. Her bruger vi det til undersøgelse. Vi undersøger om indholdet af kategori er lig strengen civil. I: Ja, men i ABAP gør man ingen forskel. Hvad er fordelen ved at tydeliggøre at nu gør vi det ene, og nu gør vi det andet. For hele konteksten i sætningen, så hvorfor dog? F: I PASCAL vil man også bruge den der altså kolon lig med, når man tildeler. I C++ vil man kun bruge lig med, når man	

Tid	Tale	Resultat og bemærkninger
	tildeler, men der vil man til gengæld have to lighedstegn, når man sammenligner. Man går det i nogle sprog, og i andre gør man det ikke. Eller så tror jeg næsten, at compileren vil æde det. Der er lige noget der. I: Når, ELSEIF F: Jep, lad os prøve at hoppe videre til opgave 3.	
35:45 **Opgave 3**	I: (Kopierer tekst og skriver). Integer, og så står der at det kan blive meget stort, BIGINT, så skal vi loope rundt i det. F: Det er rigtigt, for der findes ikke en fakultetfunktion i HANA databasen. I: Hvad er x så her, det er vores et eller andet, vi gætter, DO, END for, hvad er det så jeg vil, så vil jeg gange, hvilken loop? F: Du kan fint komme igennem med begge dele, så det er ikke afgørende. I: Det skal i hvert fald starte med mit input tal, og så skal jeg trække en fra, og gange, jeg skal måske nok holde fast i mit input tal, jeg skal måske lige skrive det, x lig med, kan man regne på ex_result, gange, står der x er nul og result er nul, det dur ikke, vi starter med at sætte ex_result til at være, så, nu har vi så vores femtal, så tæller vi fra et op til vores femtal, x er lig med vores, nej ex_result er lig med vores femtal gange vores femtal, det du ikke, så vi skal trække det her fra, femtal, så trækker vi en fra vores oprindelige femtal, så siger, så skal vi tage det der er nu femtal + fire, det gør vi fem gange, undervejs tæller vi ned, risikoen er at vi kommer ned på nul. Hvis tallet er to, så bruger jeg bare while, og im_tal større en nul DO im_tal gange 1, der er stadigt en, så er vi nede på nul, så vil den ikke gå igennem, to gange en det er to, så siger den nul, hvis vi skal gange med ettal, det behøver vi ikke gøre. F: Bare en enkelt kommentar: fem gange fire gange tre gange to gange en, det er det samme som en gange to gange tre gange fire gange fem. I: Ja. F: Det med at tælle baglæns, og styre hvornår man rammer	I Mangler BEGIN-END i sin kode. Logikken i funktionen driller I. Det indikerer at I er lidt usikker i imperativ programmering. I glemmer flere gange kolon foran variabler der refereres. I bruger kolon foran variabel, hvor der ikke refereres.

Tid	Tale	Resultat og bemærkninger
	nul, havde måske ikke været nødvendigt. Men skidt med det. Lad os prøve at hoppe videre til opgave 4. Jeg tror godt at det der kunne køre.	
48:19 **Opgave 4**	F: Du skal nok lige hæfte dig ved det, der står nede i de tre linjer her. Det er en tabel, der er lavet i forvejen. I: (Læser opgaven og mumler) F: Nu har du ikke arbejdet med HANA database, men den kan deles op i flere schemaer, og så kan kode lægges i schemaet, det gør man i repository. Så når du finder det rigtige eksempel i cheat sheet, så vil du nok opdage, at du får behov for et scheme og repository. Og der har du så navnene på de to og derudover så har du også navnet på tabellen. Og husk også at du har datatyperne på kolonnerne i tabellen. Så i den her øvelse, skal du bladre lidt frem og tilbage. I: (Læser) Jeg skal finde dem der har den titel. F: Ja, og hvad er output? I: En tabel. F: Ja, en tabel, hvor du tidligere bare har ført et tal ud. I: Ja, så funktionen skal starte med noget med en tabel. F: Du skal kigge heroppe og se, hvad den returnerer. (mumler). I: Nu prøver jeg ikke at klippe klistre, (skriver og mumler), og så er det mit funktionsnavn, som jeg selv bestemmer, sådan, og så en variabel, det er input, og det var en titel, hvad skal jeg så have, en titel, og der specificerer jeg de tre kolonner, det er de tre kolonner, der er navn, sådan, kategori, og timeløn int, okay. Language, det må… F: Der kommer to linjer, som du egentligt bare kan skrive af. Den første fortæller HANA databasen, at det der kommer her skal tolkes om SQL Script. Og det næste siger at når den kører det her kald, skal det køres med de rettigheder den bruger, der kalder, har. I: Yes, jah. Så skal vi slå op i den tabel der. (skriver og mumler) SELECT, timeløn, (skriver), titel, nu kaldte jeg den titel, det er nok ikke så godt…	I vælger ikke at kopiere tidligere kode, men at skrive det hele meget støttet af cheat sheet. På den måde har I tid til at tænke undervejs. I er lidt usikker på brugen af schema og repository.

Tid	Tale	Resultat og bemærkninger
	Ja, jeg er i tvivl om det skal med. F: Ja, det skulle jeg til at spørge dig om. I: Fordi vi schemaet, repository kolon kolon og funktionsnavnet. F: I schemaet ligger både tabeller og repository, hvor man kan lægge kode ind i. I: Så repository er til kode – ikke til tabeller. F: Lige nøjagtig. I: Right, kolon her. F: Jeg tror, at det vil køre. Har du en fornemmelse af hvad ordet RETURN betyder her. Jeg tror ikke, at du kan finde det noget sted i cheat sheet, andet end at du kan se det skal være der? I: Jeg kan ikke rigtigt bidrage med noget. F: Det kan jeg godt forstå, men heroppe definerer du, at det der kommer ud bliver en table, og hernede fortæller du, at nu vil du til at fortælle, hvad den table skal indeholde. Og så kommer dit SQL statement. I: Der vil vi ABAP skrive SELECT INTO. F: Ja, det vil man typisk, i hvert fald hvis man vil fylde noget ind i, en ekstra række ind i en tabel. Okay. Prøv at læse opgave fem.	
1:05:40 **Opgave 5**	F: Du skal ikke begynde at kode, det tror jeg ikke at vi når. Jeg kunne godt tænke mig, når du bare læser de første fire fem linjer, at høre om du kan se hvordan den opgave adskiller sig fra den, du lige har lavet. I: Jeg skal selv generere det, der skal ned i tabellen. Det kommer ikke noget sted fra. F: Ja, det er rigtigt, tabellen ligger i forvejen i databasen, og du skal så fylde data ned i tabellen. Hvor du heroppe blot lavede en forespørgsel, der fik data tilbage, som du kunne arbejde videre med. Her skal du faktisk manipulere med data på databasen. Hvis vi havde haft lidt mere tid, ville du	

Tid	Tale	Resultat og bemærkninger
	have opdaget, at det kan man ikke gøre på samme måde. Her skal vi bruge stored procedures. Det slipper du for den her gang, men det kan være, at du bliver ramt af det i den virkelige verden. I: Ja det gør jeg nok.	
1:07:13 Afslutning	F: Vi har talt om syntaksen under vejs, herunder hvad det ligner, og hvordan man gør i ABAP. Er der noget, hvor du tænker, at det ligner noget du kender meget, og her er noget som ikke rigtigt ligner, her er noget som er irriterende, eller her kunne det have været rart at have haft hjælp af et ordentligt udviklingsværktøj, der lige kunne have understreget… I: Altså SELECT FROM WHERE er ret generisk. RETURN er smart – at kunne sige at vi vil have det ind i en returtabel. Så det er anderledes, men når man lige fanger hvorfor… F: Hvad tænker du om brugen af kolonner rundt omkring? I: At dem har jeg ikke gennemskuet nødvendigheden af. Konteksten giver ligesom forskellen. F: Det er meget præcist sagt, at det i forvejen fremgår af konteksten. I: I ABAP sætter vi punktum efter hver eneste linje, her sætter vi så semikolon – nogen steder. F: Ja, de fleste steder, der hvor man er færdig med noget. Ved IF er der ikke, det bliver fulgt om af THEN. Det tror jeg heller ikke man gør i ABAP. Sætter man punktum efter IF. I: Ja, stort set alt. F: Hvad det angår, er det her helt magen til C++ eller C#. Hvornår har du skrevet din første linje kode? Firserne? I: Ja, ja. F: Halvfjerdserne? I: Ja, det tror jeg faktisk, nej det må have være i 81, tror jeg. Da jeg var færdig med min første omgang i forsvaret og startede på HF, der lånte jeg en Commodore et eller andet. Da skrev jeg basic.	

Tid	Tale	Resultat og bemærkninger
	F: Det var noget skrækkeligt noget med linjetal foran hver linje, og mange bruge pænt 10, 20 så der var plads, og GOTO, når der ikke var plads. Det var et helvede at renummere det hele. GOTO til linjenumre. Så har du kodet jævnligt lige siden. I: Ja, i starten var det ren sjov. Så kom jeg på universitetet hvor det var FORTRAN. Så fik jeg job. Der var det COBOL. F: FORTRAN og COBOL har jeg aldrig prøvet. Har du været i banksektoren, det er vel typisk i økonomiprogrammer, der bruges COBOL. I: Det var hos Dronningborg, der laver mejetærskere. I deres IT afdeling. Hele deres ERP system var hjemmelavet i COBOL….	

Resulterende kode:

```
create function convert_pound (im_pound dec (10, 4))
returns ex_result dec (10, 4) as
begin
  ex_result := :im_pound / 2.2046;
end
create function calc_overarbejde (kategori varchar(30), im_timeloen INT)
returns ex_result dec (10, 2) as
begin
if :kategori = 'Militær' then
  ex_result := :im_timeloen * 1.4;
elseif :kategori = 'Civil' then
  ex_result := :im_timeloen * 1.6;
else
  ex_result := :im_timeloen;
end if;
end
```

```
create function calc_fak (im_tal dec BIGINT)
returns ex_result BIGINT) as
ex_result := im_tal;
  while im_tal > 0 do
  :im_tal := im_tal - 1;
  ex_result := ex_result * :im_tal
End while;


function "FORSVARET"."FORSVARET_PAC::TITEL_F" P_Titel VARCHAR(30)
Returns table (navn VARCHAR(30),
                      Kategori VARCHAR(30),
                      Timeloen INT)
LANGUAGE SQLSCRIPT
SQL SECURITY INVOKER AS
BEGIN
  RETURN
  SELECT Navn, kategori, timeloen
  FROM "FORSVARET"."Personel"
  WHERE Titel = :P_titel;
End;
```

Transskribering af forsøg med IBM medarbejder nummer 6.

Fil: 2019-10-04_11-11-35.mp4

Gennemført 2019-10-04.

Tid	Tale	Resultat og bemærkninger
29:50 Indledning.	F: Om lidt så stikker jeg dig et cheat sheet, og det det går ud på er SAP HANA SQL script. Jeg ved ikke, om du har læst det, som Martin på et tidspunkt sendte rund? SAP HANA150. I: Jooh… F: Hvis du har læst det, og kan huske det, er du godt kørende. Hvis ikke du har læst det eller kan huske det, så kunne du nok have fornøjelse af at kigge lidt på det her (cheat sheet). Om lidt får du fem minutter til at kigge på det. Du skal ikke kigge på alle detaljer. Prøv at få en fornemmelse af hvad du cirka kan finde på de to sider. Tre kolonner på hver side. Det får du fred og ro til. Så jeg sætter mig lige hen og laver andet. Når du har gjort det, så kommer jeg tilbage. Så tager vi stille og roligt opgave sheet, og så får du lov til at hygge dig med det, imens du tænker højt. Og vi snakker om det, og nogle gange så peger jeg på skærmen og siger fx, du har skrevet det, kan du ikke forklare mig hvorfor. Det kan også være, at du siger, at her kan det løses på to forskellige måder, så kan det være at jeg siger, at det kan være lige meget, gå bare den vej. Du skal ikke forvente, at du bliver færdig, det værste der kan ske er, at du løser alle opgaverne uden fejl snip snap, for så får jeg ingen ting med. Det jeg er interesseret i, er at se, hvor driller det. Så det gør ikke noget hvis du nogen gange kører fast. Du ved heller ikke hvornår, du er færdig, for du har fået det her lækre udviklingsværktøj – Wordpad. Der er ingen hjælp at hente. Så på et eller andet tidspunkt siger jeg, at nu hopper vi videre. Hvis der opstår noget under vejs, så tager vi det bare. Jeg skal lige se, om jeg har fået startet Snagit. Ja, den kører. Den ligger dernede og blinker – den optager. Så nu får du det her at hygge dig med. Fem minutter eller som det nu passer. Så fortsætter vi.	
37:30 Opgave 1.	F: Nu skal du se, her er opgaverne. Den første opgave, det er hernede. Så står der noget heroppe, som er relevant for nogle af de næste opgaver, vi kommer til. Det behøver du ikke for at løse den første. Det er sådan at sværhedsgraden stiger lidt efter den første, du skal ikke forvente at nå det hele. Prøv lige at læse den første opgave.	I følger eksemplet i cheat sheet tæt. I definerer input som INTEGER. Adspurgt er I helt klar over konsekvensen.

Tid	Tale	Resultat og bemærkninger
	I: Det bliver spændende. Det er mange år siden. Der bliver puttet en mængde ind i x antal kilo, og så skal vi have divideret det med… men.. jeg det passer ikke. Der kommer 7 komma 3, det er x ind, så skulle vi gerne dividere med 2 komma 2046 og så skulle vi gerne få 3 komma 3 kilo. F: Så vi skal have fundet noget, der hjælper os at få en parameter ind og spytte et tal ud. I: Så det var det. F: Det er med lige at finde det rigtige sted i cheat sheet. Det er ikke så afgørende at lede. Du skal herom et sted og så herop i nærheden. Her har du user defined function… I: Ja, den kommer ind convert hours, det ligner et eller andet, det skal bare ikke være hours, og den returnerer en eller anden værdi. Som kommer ud gange tres. Så de tres må være den der, der skal bare divideres. F: Lige præcis, så du har et skelet. I: Jeg begynder bare at taste. Det er mange år siden, der er ikke stavekontrol eller anden hjælp. (skriver og mumler). Nu vil jeg gerne teste det. F: Det vil nok næsten kunne køre. Du mangler nok et enkelt semikolon. I: Ja, det gør der til sidst. F: Hvis du er vandt til ABAP, så det et punktum. I: Jeg læser jo kun, hvad de andre har skrevet. F: Det tror jeg godt at ville kunne køre. Det er ikke sikkert, at det vil reagere helt, som du gerne vil have, men det kan vi lige snakke om. Jeg kunne forestille mig, at jeg godt kunne komme med tre et halvt pund og så have det regnet om til kilo. Kan du rutine eller funktion håndtere det? I: Det vil nok give en udfordring. For det der er INTEGER, det er et helt tal, så det skal være noget andet. DEC den er der. F: Ja, og så skal du angive noget efter DEC, ligesom du nedenunder har fem komma fire, så skal du nok finde ud af et eller andet.	Er usikker på brugen af antal cifre og forveksler det med tallets størrelse.

Tid	Tale	Resultat og bemærkninger
	I: Så den skal have det samme. F: Måske, du skal nok overveje, hvad fem og fire betyder. Du kan få lidt hjælp i cheat i sheet. I: Nå, total og fraction. Der tror jeg, at jeg melder pas. F: Total, det er hvor mange cifre tallet må bestå af og fraction er, hvor mange der må være til højre for kommaet. Her siger du, at det må være et fem cifret tal, og der må kun stå et ciffer til venstre for kommaet og fire cifre til højre for kommaet. Så den kan kun omregne til noget, der er mindre end 10 kilo. Så det kan være, at du skal afsætte fx 10 cifre og to efter kommaet. I: Okay, så den skal skrues op. Nu har vi 99 cifre, det er et stort tal, bare ni er et stort tal. F: Du kan overveje det samme hernede (i output). I: Den notation kunne jeg ikke huske. F: Jeg synes, at du skal lave et par blanke linjer og skrive opgave 2. Jeg optager alt, hvad du laver, med Snagit.	
47:50 **Opgave 2.**	I: (Læser) Okay. Det første vi skal finde ud af er selve tabellen. Den har et tabelnavn. Det skal vi tage udgangspunkt i. Når vi så har den rigtige tabel, så skal vi have fundet kategorierne, som bare hedder kategori. Der er to muligheder militær og civil, nej der er også blank. Der var en præst, han er hverken det ene eller det andet. Så der er tre kategorier, dem skal vi vælge imellem. Hvis det er en militær, så skal der ganges med en komma fire. Hvis det er civil så en komma seks, alt andet så skal der ikke gøres noget. Det var det simple på papiret, så kommer udfordringerne. F: Men det er sådan set ikke så svært. Du skal ikke ind og læse i tabellen, du skal bare fra tabellen tage input fra tabellen. Det interessante er, at du får en kategori og en timeløn, før fik du kun en vægt i pund. Nu får du to inputparametre. Hvis du kigger her, så har du en user defined function med to input. I: Det er jo fantastisk – og den gode gamle if sætning. Vi har en god start. (Skriver og mumler) lønnen er i hele kroner.	F hjælper for at komme godt i gang med opgaven. I er usikker på brugen af datatyper. I bruger INTEGER som output fra funktionen, og ”definerer” INTEGER som (3,0). I følger eksemplet i cheat sheet meget tæt. I er usikker på bruger af DEC(t, f). I forstår ikke forskellen på lig med og kolon lig med.

Tid	Tale	Resultat og bemærkninger
	Så er der lige alle parenteserne. Hvor blev RETURNS af – der skulle gerne komme noget ud. Så tror jeg, at vi er klar til at teste. F: Prøv at løbe det igennem. I: Vi laver en funktion, som går på timelønnen, som er hele tal, og det skal omdannes til et beløb. F: Det er din kategori, du får ind her. Hvis kategorien er militær, skal du gange med en komma fire, hvis den er civil skal du gange med en komma seks. Du har sat kategorien til at være to karakterer, altså to bogstaver. I: Det er nok i underkanten. Der står jo også 30. Jeg kunne jo bare have kigget efter, der står jo 30. F: Og så er det rigtigt, at når du får et helt tal ind og ganger med en komma fire eller en komma seks, så bliver det ikke et heltal, der kommer ud, så du har fat i, at du har noget her… I: Nå ja, det er rigtigt, det bliver selvfølgeligt et decimaltal, det kommer an på, hvad man vil. Hvad står der i opgaven. F: Både de civile og soldaterne vil gerne have de sidste ører med. I: Jeg kunne jo sige, at opgaven ikke definerer det. Men det er rigtigt… F: Hvad kan den rumme nu, hvis den er tre komma to? I: Ikke så meget. F: Den kan rumme næsten 10 kroner, det er en lidt lav timeløn, selv om vi er offentligt ansatte. Så kommer der nogle betingelser. I: Der kunne man godt være fræk, det oplever vi nogle gange i den virkelige verden. Det kræver, at det er skrevet helt nøjagtigt. Der kunne man godt på en eller anden måde, som jeg ikke lige kan huske i hovedet, sætte op at det er ligegyldigt, om det er skrevet med stort eller småt for at tilsikre at man rammer hvad som helst. F: Det kunne også være i inputsystemet, at der lave feltvalidering. Det kunne fx være en rullebox.	

Tid	Tale	Resultat og bemærkninger
	I: Det er bare meget kendt fra dagligdagen, hvor vi kæmper med den slags. F: Hernede har du en ENDIF, hvor du starter du din IF? I: Det gør jeg sådan set ikke rigtigt nogen steder. Lige nu. Den var der, en trykfejl. Det er rigtigt. F: Her bruger du lighedstegn, der står alene, hernede bruger du kolon lighedstegn. Har du en fornemmelse af hvorfor? I: Nej faktisk ikke, hvis jeg skal være helt ærlig. F: Når du bare sætter lighedstegn, så er det fordi du sammenligner. Du sammenligner, hvad står der i den variabel med ordet militær. Hernede laver du en tildeling med kolon lig med, så du siger, det der står her gange en komma fire tildeles over til export result. Det er rigtigt skrevet, bortset fra hernede, der er der et lighedstegn for meget. Når du bare har kolon foran en variabel som der og der, så betyder det, at nu er det ikke variablen, du er interesseret i, det er værdien der står i variablen. Jeg tror, at det vil køre igennem nu. I: Det tror jeg også, men jeg ville ikke lave den der lig med. Jeg ville sikre mig, at man kan skrive med stort og småt. Det er en teknikalitet, et spørgsmål om hvor tillid man har. (En kort drøftelse af validering). Lad os prøve at hoppe videre til opgave 3.	
1:04:20 **Opgave 3.**	I: (Læser) Okay. Rent matematisk er det rimeligt forståeligt. Man putter et vilkårligt tal ind, og så vil man gerne have at det ganges mens man tæller nedad. Så stod der eksempelvis tre her, så er det tre-to-en. Så rent matematisk er det forståeligt. F: Hvor mange input har du? I: Jeg har kun en. Det tal der tastes ind. Så tricket er, eller der er to ting. Hvis det er et stort tal, så bliver resultatet et meget stort tal, og det der skal gange bliver til mange parameter. 10 giver fx allerede et meget stort tal. F: En parameter ind og et tal ud. Det minder om en af de opgaver, vi har kigget på. I: Ja det gjorde vi jo i den første, der puttede vi i hvert fald et	I definerer INTEGER(9,0). I er altså klar over, at der ikke er decimaler på en INTEGER. I har problemer med at omsætte logikken til imperativ kode. I lyder til at have de ”rigtige” tanker, men har svært ved at omsætte dem til imperativ kode. Opgaven ender ikke meget sammenhængende kode.

Tid	Tale	Resultat og bemærkninger
	tal ind, så det kan vi begynde med. Må jeg kopiere? F: Det må du gerne, det er der også mange af dine kollegaer, der har gjort. I: Så undgår jeg også slåfejl. Så det, vi skal have, er en funktion. Vi skal have konverteret et eller andet tal. Det hedder x. (Skriver) Det kan af gode grunde ikke blive et decimaltal, og heller ikke et negativt. Godt, så skal den komme ud med et eller andet. Det er også et helt tal af en art. Så kommer det spændende, hvordan skal den regne det ud. Den begynder med x, og så tæller den ned. (Læser) Loops. Det må være den del vi skal over i. Loop as long as, men den skal jo gange. (Skriver og mumler). Det her kan nogenlunde gå, kan jeg regne ud, så længe vi har hele tal, men på et tidspunkt bliver den nul. Den bør jo smage på, om vi er kommet til nul, på en eller anden måde. (Skriver) Det vi siger er, at så længe den er over nul, så skal den regne, på det tidspunkt den bliver nul, så skal vi bare have et resultat. Det er egentligt det, der skal til. Det ville jeg så bestille en programmør til. F: Det er du også meget velkommen til, du har bare begge kasketter. I: Ja, det er så det der, der skal køres rundt, men den skal selvfølgelig også huske det, på en eller anden måde. Den skal i hvert fald huske, at den har regnet ud, at det her er tyve. F: Hvis du har behov for en variabel, kan du skrive DECLARE ligesom her. Så hvis du har behov for en mellemregning… I: Jeg har jo brug for, at den husker, at det er tyve. (Skriver og mumler). Den skal have at vide, hvad der skal ske, hvis den er nul. Når den er nul. Det er bare spørgsmålet, hvor det skal så henne. For når den er nul, så er vi klar til at få et resultat. Men det som sagt er hensigten er, at den bare skal blive ved så længe. Vi har puttet et x ind, så skal den bare fylde op, huske det, og når x er nul, så kan vi ikke regne mere, for hvis vi ganger med nul, så ødelægger vi det hele. F: Og så var tanken, at når vi når nul, så skal den levere resultatet. Det der, det vil nok ikke køre. Men det er fint nok, jeg har set det, som jeg kigger efter. Jeg synes, at vi skal gå videre til opgave 4. Og prøve en anden type opgave.	

Tid	Tale	Resultat og bemærkninger
1:19:05 **Opgave 4**	I: (Læser) Okay, så har vi fat i den der igen. Denne opgave er modsat. F: Ja, prøv at fortælle, hvordan den adskiller sig fra de andre. I: Det vi kigger på er den kolonne der. Hvis der fx er to kaptajner, så vil vi gerne have dem valgt ud, og derefter vil vi gerne have deres navne, kategori og timeløn. F: Hvad skal funktionen returnere? I: Den skal returnere navn, kategori og timeløn i en tabel. F: Præcis, det vi har set indtil nu, der fik man en værdi ud. Nu skal du finde et eller andet, der giver en tabel. Det er nok den første udfordring, at finde noget i cheat sheet, der giver en tabel. I: Der var et eller andet med user table. Det var den der, hvor man starter med en tabel. F: Inden du går det, så tror jeg, at du lige skal læse de par linjer, der står nedenunder tabellen. I: Nåh ja, der er et schema navn. F: Så det er en tabel, der ligger i vores HANA database. Du kender det med schemaer. I schemaet har vi et repository, hvor vi lægger vores kode. Så i denne her HANA database, der har vi et schema, der hedder forsvaret, og i det schema har vi forsvaret underscore pac som repository. Når du ser det her ovre, så vil du se, at du skal bruge schemaet og repository på et tidspunkt. Der har du skelettet. I: (Skriver og mumler). F: Det der kommer der, er det navn du har lyst til at give din funktion. I: (Skriver og mumler). (Gennemlæser koden og retter) F: Det begynder at ligne noget. Jeg har et par spørgsmål. Hvis det der blev tastet ind i HANA Studio, hvad ville der så ske med den kode. Hvor ville den ende. Du behøver ikke rette noget. I: Det var det du nævnte i starten, det med pac. Det skulle den klare heroppe.	I ser ikke helt ud til at have forstået det smarte ved at kunne kalde en funktion, med en parameter, der indgår i SQL kaldet. I blændes muligvis af skabelonerne i cheat sheet. I fokuserer på at få koden i overensstemmelse med cheat sheet snarere end tænke over, hvad der rent faktisk foregår.

Tid	Tale	Resultat og bemærkninger
	F: Det jeg fisker efter, er det, der er særegent for HANA databasen, det at man kan lægge koden ned i HANA databasen, og så afvikle den dernede. Hvor vi som regel har en applikationsserver ved siden af, hvor koden kører, så denne kode kunne ligge direkte nede i databasen. Så hvis denne tabel var på 40 mio. records, hvis den kørte på den gamle måde, så ville vi flytte hele tabellen over på applikationsserveren, så ville vi foretage søgningen på to kaptajner, og så ville vi smide de 40 mio. records ud igen. Nu lægger vi i stedet for koden ned i databasen og afvikler den der. Så vi slipper for hele den der overførsel af data. I: Ja, så udvælgelsen foregår dernede. F: Har du en fornemmelse af, hvad den linje der betyder? I: Ja, den trækker den op på serveren og afvikler den der. F: Det den siger er, at når den afvikler dette script, så skal den bruge invorks, altså den der kalder funktionens rettigheder. Så når man beder HANA databasen om at køre det her, så kigger den på, hvem er det der kalder, og det er Benny, så vil den køre det med Bennys rettigheder på HANA databasen. Man kunne også vælge at sige, at den skulle prøve at køre det med det som programmøren havde af rettigheder. Men hvis ikke du har programmørrettigheder, så vil den komme tilbage og sige, at det dur ikke, så derfor vil man normalt bruge denne her. I: Ahh, okay. F: Du skal ikke skrive mere, jeg trykker bare lige gem.	
1:36:14 Afslutning.	F: Nu er vi nået dertil, hvor jeg skal høre, om der var noget, hvor det ligner noget, du har set før, hvor du tænkte, at det var nemt eller svært, eller at det var irriterende at man hele tiden skulle huske noget. I: Næh, for mig er det lidt en iterativ proces. Jeg starter med et eller andet, men jeg har ikke tænkt det til ende, for når jeg så kommer rundt, det er nok min måde, at det kører rundt flere gange. For mig behøver det ikke at være helt klar, før jeg prøver at køre det igennem. Så er jeg nok, skruet sådan sammen, at jeg lige skal finde logikken, hvordan jeg så får det ned på papir kan være spændende. Grundlæggende er det bare at bruge matematikken.	

Tid	Tale	Resultat og bemærkninger
	F: Det kunne jeg tydeligt høre i opgave 3, du havde de rigtige tanker, men det drillede var at omsætte det til kode. I: I den sammenhæng ville jeg tænke meget i Excel, det ville være der hvor jeg primært…, og så kan man sige, det andet der ikke lige er til rådighed er en testfunktion, hvor man lige kan få luget de værste ting ud, hvis man fx har glemt et semikolon. F: Præcis, der er ingen hjælp her, og det er helt velbegrundet. Det ligger i modellen. Du ville have fået alt for meget hjælp, hvis vi fx havde brugt HANA Studio. I: Ja, så ville man bare have højreklikket… F: Jeg tror at du sagde undervejs, at du ikke skriver kode til dagligt, men læser det, andre har skrevet. I: Ja, hvis vi har en fejl kan jeg sagtens læse det, de andre har lavet. F: Har du skrevet kode tidligere? I: Ja, men det er mange år siden. Jeg fandt ud af, at der var nogle, der var bedre til det end mig – og hurtigere. Det er også et træningsspørgsmål. Hvis du gør det, så får du træningen. Der var nogen, de var både hurtigere og bedre til det. F: Hvilke sprog har du arbejdet med? I: Algol helt tilbage og Pascal, det var der, det startede for rigtigt mange år siden, dengang vi troede, at vi kunne redde hele verdenen – men det kunne vi ikke. F: Hvad med SQL? I: Ja lidt, men det er primært også det med at læse det. Min hjerne kan godt forstå det, det er et spørgsmål om at lære det… F: Det virkede bare som om, du var lidt mere hjemmevandt med det her (SQL), end med fx WHILE. I: Ja, men det er det vi bruger meget. Og så i if sætninger. Det er mange gange der, vi ser fejl. Ting opstår fordi, der er en forkert selektering eller noget i den stil. Så det er ofte der man oftest finder et eller andet. While ser jeg ikke så tit. Det	

Tid	Tale	Resultat og bemærkninger
	er mest selektioner, der giver fejl… F: Vi er kommet igennem.	

Resulterende kode:

```
CREATE FUNCTIONS Convert_lbs (im_lbs DEC (9,2)
RETURNS ex_result DEC(9,4) AS
BEGIN
  ex_result := im_lbs / 2.2046;
END


CREATE FUNCTION Convert_Timeloen (im_timeloen INT,
           in_to VARCHAR (30) )
RETURNS ex_result DEC (5,2) AS
BEGIN
  if : im_to = 'Militær' then
      ex_result := :im_Timeloen * 1.4;
ELSEIF :im_to = 'Civil' then
     ex_result := :im_Timeloen * 1.6;
ELSE
    ex_result := :im_Timeloen;
 END IF;
END;



CREATE FUNCTIONS Convert_x (im_x INTEGER)
RETURNS ex_result INTEGER AS
WHILE : x -1 < X DO
if :in > 0 THEN x := x * (x-1)
DECLARE local_var INT
END WHILE
```

```
if :in = 0


FUNCTION  "FORSVARET" ."PAC::tfSAMLING"
                                                        title VARCHAR (30))
RETURNS TABLE VARCHAR(Navn ,VARCHAR (30),
Kategori VARCHAR (30), Timelon INT)
LANGUAGE SQLSCRIPT
SQL SECURITY INVOKER AS
BEGIN
 RETURN
 SELECT navn, kategori, Timelon
FROM "FORSVARET"."SAMLING"
WHERE titel = :Kaptajn;
END;
```