Datavetenskap

Nancy Sheik-Khalil

Edison Meneses

Enkätkonverterare

Examensarbete, C-nivå

2005:02

Enkätkonverterare

Nancy Sheik-Khalil

Edison Meneses

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Nancy Sheik-Khalil

Edison Meneses

Godkänd, 2005-01-14

Handledare: Robin Staxhammar

Examinator: Martin Blom

Sammanfattning

Prifloat System AB är ett företag som hjälper sina kunder att analysera och presentera enkätundersökningar, genom att använda sitt webbaserade verktyg Prifloat. Idag skrivs svaren från enkäterna in manuellt i Prifloat Infosystems databas, vilket är ineffektivt och tidskrävande.

Företaget Evolve har fått i uppdrag från Prifloat System AB att automatisera den manuella processen att läsa in data från enkäter och formatera svaren till att passa Prifloat Infosystems databas. Vi har fått uppgiften att skapa en enkätkonverterare som kan transformera ReadSoft Forms utdatafiler till ett format som passar Prifloat Infosystems databas.

Rapporten beskriver programvaran Readsoft Forms och hur vi analyserar de olika sätt som man kan forma en utdatafil på. Vi fick en mall från Evolve, som visar hur enkätsvaren bör se ut för att de ska kunna lagras i Prifloat Infosystems databas. I rapporten beskrivs det hur vi direkt i Readsoft Forms försöker anpassa utdatafilen att se ut enligt mallen. Rapporten visar även hur vi formar en utdatafil på enklast möjliga sätt, och med hjälp av en programmodul anpassar utdatafilen att se ut enligt mallen.

Resultatet blir en programmodul i PHP som läser en utdatafil från Readsoft Forms och genererar utdata som följer mallens utseende.

Abstract

Prifloat System AB is a company that helps their customers to analyze and present questionnaires. This is done through the use of their web based tool Prifloat. Today, the replies from the questionnaires are typed in manually into the Prifloat Infosystems database, this is ineffective and time-consuming.

The company Evolve has been commissioned, by Prifloat System AB, to automatize the process of manually reading and typing data from questionnaires and to format the replies to fit the Prifloat database. We got the task to create a questionnaire transformer that can transform ReadSoft Forms output files to a format that fits the Prifloat Infosystems database.
The report describes the ReadSoft Forms software and how we analyze the different ways in which an output file may be formatted. We got a template from Evolve, which shows how the questionnaire results should look in order to be stored in the Prifloat Infosystems database. In the report we describe how we try to adapt the output file, in Forms, to the template. The report also shows how we format an output file in the simplest way possible, and with the aid of a program module adapt the output file to conform to the template.

The result is a program module, written in PHP, which reads an output file from Forms and generates an output file that conforms to the template.

Innehållsförteckning

1 Inledning 1

1.1 Bakgrund 1

1.2 Mål 1

1.3 Disposition 2

2 Readsoft Forms 3

2.1 Forms 3

2.2 Manager 3

2.2.1 Formulärdefinition 4

2.2.2 Transaktionsbeskrivning 7

2.2.3 Jobbeskrivning 9

2.3 Scan 10

2.4 Interpret 10

2.5 Verify 10

2.6 Transfer 11

3 Prifloat 13

3.1 Prifloat Infosystem 13

3.2 Mallen 15

4 Analys och anpassning av utdatafiler 17

4.1 Forms utdatafil 17

4.1.1 Två alternativ att presentera utdatafil på 17

4.1.2 Analys av de två alternativen 20

4.2 PHP 23

4.2.1 Varför PHP? 23

4.2.2 Vilken definition har bestämts 24

5 Implementering 29

6 Resultat och rekommendationer 33

6.1 Resultat 33

6.1.1 Studera Forms 33

6.1.2 Studera Prifloat Infosystems databas 33

6.1.3 Analysera och anpassa utdatafiler 33

6.1.4 Lagra utdatafil i databasen 34

6.2 Rekommendationer 34

7 Summering av projektet 35

Referenser 36

A Bilaga 37

Figurförteckning

2.1 Exempel på en enkät. 5
2.2 Komprimerad utdatafil med tabbtecken som avskiljare..... 8
2.3 Utdatafil med kommatecken som avskiljning och utan komprimering. 8
2.4 Utdatafil med rubriker. 8
3.1 Visar hur databas och webbserver interagerar. 14
3.2 Utseendemall för indata. 15
3.3 En enkät med frågor av typen betyg. 16
3.4 En enkät med frågor av typen index. 16
4.1 Exempel på en enkät, en formulärdefinition, en transaktionsbeskrivning och en utdatafil. 18
4.2 Exempel på en enkät med unikt definierade värden. 19
4.3 Exempel på en enkät, en formulärdefinition, en transaktionsbeskrivning och en utdatafil. 20
4.4 Exempel på vad som ska göras för att få utdatafilen att följa mallen i kapitel 3. 22
4.5 Enkät med textfrågor. 24
4.6 Betyg - och indexfrågor läggs in. 25
4.7 Kategori och alternativ frågor läggs in. 26
5.1 Utdatafil som följer vår tabell i kapitel 4. 29
5.2 Arrayen `delar[]`, efter att funktionen `split()` utförts på utdatafilen. 30
5.3 Utseendet på den nya arrayen `nya[]`. 32

Tabellförteckning

2.1 En tabell som redovisar fältens innebörd 9
4.1 Visar vad vi har valt att tilldela varje frågetyp. 28

1 Inledning

1.1 Bakgrund

Företaget Evolve Development AB på Inovahuset arbetar med webbsystem med teknisk fokus för Internet. Det mesta från enkla hemsidor till större webbaserade applikationer och webbhotellslösningar erbjuds. Evolve har tidigare utvecklat den tekniska plattformen som utgör Prifloat Infosystem och har fått förtroendet att ansvara för drift och utveckling av systemet.

Prifloat System AB är ett företag som hjälper sina kunder med att presentera sina enkätundersökningar, genom att använda sitt webbaserade verktyg Prifloat. Med Prifloatverktyget samlas svaren från enkäter in, analyseras och presenteras med exempelvis tabeller och diagram. Idag läses de papperstryckta enkäterna in manuellt, genom att mata in en enkät i taget för hand lagras svaren i en databas där de sedan kan undersökas. Detta är både ineffektivt och tidskrävande.

Evolve har nu fått i uppdrag från Prifloat System AB att med ReadSoft Forms automatisera den manuella processen att läsa in data från handskrivna enkäter och formatera svaren till att passa Prifloat Infosystems databas. Vi har därför fått uppgiften av Evolve att skapa en konverterare som kan läsa in ReadSoft Forms utdatafiler till Prifloat Infosystems databas.

ReadSoft Forms är ett program som läser in handskrivna eller maskinellt skrivna enkäter och på ett automatiserat sätt analyserar informationen som sedan kan överföras till en utdatafil. Utdatafilen beskriver och sammanställer den överförda utdatan, från de ifyllda enkäterna, som vi sedan skall anpassa till Prifloat Infosystems databas där data skall lagras och analyseras.

1.2 Mål

Målet är att skapa en enkätkonverterare som kan transformera ReadSoft Forms utdatafiler till ett format som passar Prifloat Infosystems databas. För att uppnå målet måste följande delmål utföras.

- **Studera Readsoft Forms** för att kunna förstå och använda programvaran vid inläsning av olika typer av enkäter och se hur data från enkäterna kan presenteras i utdatafiler. Evolve vill ha en generell lösning för alla typer av enkäter, så att utdatafilerna täcker de typer av enkäter som Prifloat kan tänkas använda.
- **Studera Prifloat Infosystems databas** för att förstå dess uppbyggnad och se hur den vill ha utdatafilerna formaterade.
- **Analysera utdatafilerna** som produceras utifrån enkäterna i Forms, för att kunna generera utdatafiler i format som är bäst lämpade för Prifloat Infosystems databas.
- **Anpassa utdatafilerna** till ett format som kan användas för insättning i Prifloat Infosystems databas.
- Om tid finns ska vi också: **hitta ett sätt att lagra datan från utdatafilerna i databasen**.

1.3 Disposition

- Kapitel 2 beskriver Readsoft Forms och dess fem olika moduler. Här ges också en beskrivning av hur vi har använt programvaran och de problem vi har stött på.
- Kapitel 3 beskriver översiktligt hur Prifloat Infosystems fungerar och beskriver även en mall som utdatafilerna ska följa.
- Kapitel 4 beskriver analysen och anpassningen av utdatafilerna.
- Kapitel 5 beskriver implementationen över programmodulen och hur utdatafilerna formateras enligt den mall vi ska följa.
- Kapitel 6 beskriver hur de olika delmålen har uppnåtts och rekommendationer över uppgiften.
- Kapitel 7 är en summering av projektet.

2 Readsoft Forms

I detta kapitel presenteras programvaran Readsoft Forms, det är programvaran som kommer att användas vid inläsning av enkäter och generering av utdatafiler. Detta kapitel kommer att ge läsaren den kunskap som behövs för att förstå hur inläsningen av enkäter sker samt hur vi skapar utdatafiler. I avsnitt 2.1 beskrivs själva programvaran och dess fem olika moduler översiktligt. I avsnitt 2.2 ges en beskrivning av den första modulen, Manager. Avsnitt 2.3 beskriver modulen Scan, avsnitt 2.4 beskriver modulen Interpret, avsnitt 2.5 ger en beskrivning av modulen Verify och avsnitt 2.6 beskriver den sista av de fem modulerna, Transfer.

2.1 Forms

Readsoft Forms är ett program som används för automatisk inläsning och hantering av information från enkäter. Programmet läser in, tolkar och verifierar informationen och överför den sedan till en utdatafil eller ett målsystem. Processen genomförs stegvis med de fem huvudmodulerna i Forms: Manager, Scan, Interpret, Verify och Transfer. Modulerna är fristående från varandra och kan antigen köras enskilda eller tillsammans på en dator eller i ett nätverk. [1]

Den första modulen Manager används för att förbereda Forms innan enkäter kan bearbetas, här ställs formulärinställningarna in för de andra modulerna. I modulen Scan läses enkäter in via en scanner och sparas som bilder för vidare hantering. Modulen Interpret tolkar informationen från bilderna av enkäterna som Scan har sparat, mot förvalda inställningar i Manager. Om informationen från enkäterna innehåller ofullständiga data eller fel som upptäcks av modulen Interpret, verifieras denna information av modulen Verify. Slutligen överförs informationen till en utdatafil eller ett målsystem av modulen Transfer.

2.2 Manager

En presentation av uttrycken formulärdefinition, jobbeskrivning och transaktionsbeskrivning kommer att ges i detta avsnitt, det är uttryck som är viktiga för att förstå hur Forms fungerar vid bearbetning av enkäter.

Det är med Manager hela det grundläggande arbetet, som låter oss lägga in nya enkäter i systemet, sker. Det finns tre steg man utför i manager vid utformning av en enkät. Det första

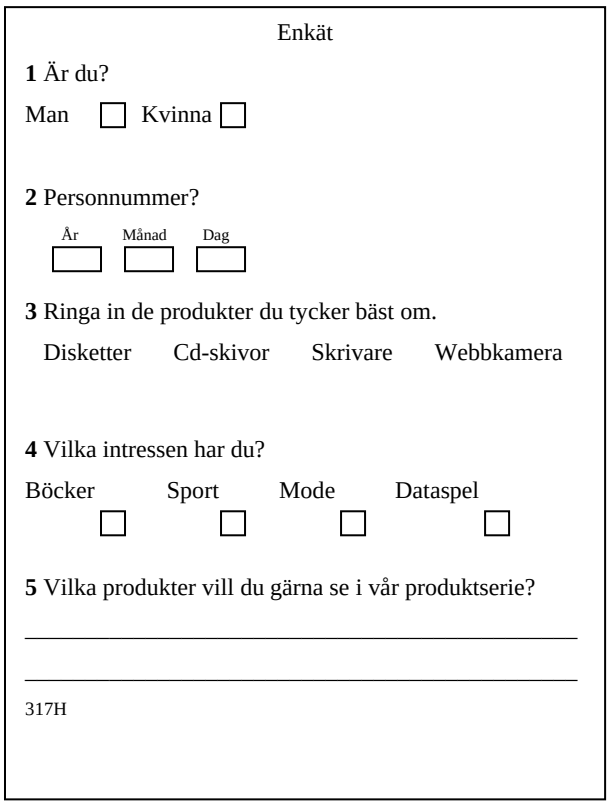
som görs i Manager är att man scannar en enkät för att kunna beskriva vilka typer av svar den består av, hela den här beskrivningen lagras i det som heter formulärdefinition, som kommer att beskrivas utförligare i avsnitt 2.2.1. Det andra steget är att skapa en utseendemall för hur de olika svaren från enkäterna ska lagras i den slutgiltiga utdatafilen som genereras av Forms, en sådan mall lagras i en transaktionsbeskrivning som kommer att beskrivas i avsnitt 2.2.2. Det sista steget är att skapa en jobbeskrivning, som beskrivs utförligare i avsnitt 2.2.3. Jobbeskrivningen är en viktig del i Manager eftersom det är den som anger hur de andra modulerna ska arbeta när de används för en enkät.

2.2.1 Formulärdefinition

När man vill jobba med en eller flera likadana enkäter måste man först beskriva en enkät för Forms, och det gör man i en formulärdefinition. Formulärdefinitionen är till för att Forms ska veta vilken typ av information som en enkät kommer att bestå av och hur den informationen ska bearbetas. Förutom frågor och svar kan en enkät bestå av annan information som kan användas till att identifiera enkäten, som exempelvis en kod. Längre fram visas exempel på information som används till att identifiera enkäter.

All information i en enkät som ska redogöras i formulärdefinitionen ramas in i en ruta som kallas fält för att fastställa typen informationen är av, och definieras i något som heter fältdefinition. Fältdefinitionen som finns i formulärdefinitionen innehåller information om var fältet befinner sig i enkäten, vilken typ av information som finns i fältet och hur det ska bearbetas av Forms fem moduler [1]. Själva definitionen av varje enskilt fält sker på så sätt att man namnger fältet och anger vad det är för typ av information som kommer att stå i fältet, informationen om fältets placering i enkäten sätts automatiskt. Det finns sju olika typer av fält som används i Forms och dessa är följande:

- Justeringsfält.
- Igenkänningsfält.
- Teckenfält.
- Markeringsfält.
- Urvalsfält.
- Bildfält.
- Streckkodfält.



Figur 2.1: Exempel på en enkät.

När man arbetar med olika enkäter i Forms måste programmet på något sätt skilja på enkäterna, därför finns ett fält som används för just det syftet och som alltid måste anges i varje enkät. Detta fält heter justeringsfält och anges för att Scanmodulen ska kunna urskilja enkäter från varandra genom att leta efter ett mönster som fälten bildar, fälten måste vara unikt valda i varje enkät. Fälten man väljer som justeringsfält kan vara vilka tecken och symboler som helst i enkäten, men får senare inte väljas som ett annat fält. Det går alltså inte att välja någon av bokstäverna i *Disketter* i figur 2.1 som ett justeringsfält eftersom det senare ska användas som ett av de andra ovannämnda fälten. Däremot går det bra att exempelvis välja siffran *2* i andra frågan i enkäten. Man kan högst välja fem justeringsfält och dessa kan befinna sig vart som helst i enkäten med restriktionen att de inte ligger för nära varandra.

Igenkänningsfält används också till att urskilja enkäter. Till skillnad från justeringsfält används igenkänningsfält för att skilja på identiska enkäter som tillhör en och samma bunt, det kan vara ett namn eller en kod som urskiljer en enkät från en annan. Man måste däremot inte ange igenkänningsfält, vilket är fallet med justeringsfält. Igenkänningsfält anges endast om varje kod har någon betydelse för senare tillfälle när svaren har sammanställts, exempelvis

om en kod i en enkät uppger personen som har fyllt i den. I figur 2.1 finns ett igenkänningsfält, koden *317H* längst ner till vänster.

Teckenfält används för fält som består av tecken, de kan vara numeriska, specialtecken, blanksteg, gemener och versaler, dessutom kan man ange om tecknen ska vara handskrivna eller maskinellt skrivna. Den andra frågan i figur 2.1 är ett teckenfält, här väljs numeriska tecken. När man definierar ett teckenfält måste man ge ett mer specifikt format på informationen. Exempelvis, i den första rutan *År* sätts format specifikationen till 2 numeriska siffror, man behöver inte ange hela årtalet det räcker med årtiondet. Om någon skriver 1 siffra eller får plats med 4 siffror i denna ruta kommer det att tolkas som ett fel av modulen Interpret. Användaren av programmet måste då åtgärda felet i modulen Verify. Rutan *Månad,* kan bestå av ett värde mellan 1 och 12, eftersom det endast finns 12 månader, och kan därför vara ett – eller tvåsiffrigt. Om någon skriver 13 i den rutan kommer modulen Interpret att tolka det som ett fel, och man får åtgärda det i modulen Verify. Även i rutan *Dag* anges format specifikationer som också är ett – eller tvåsiffrigt. Man specificerar format i teckenfält för att förenkla och förbättra tolkningen i modulen Interpret.

Markeringsfält används för de fält som består av rutor som markeras med ett kryss, man kan bestämma om ett flertal rutor hänger ihop till en fråga eller om frågan endast har en tillhörande ruta. I figur 2.1 har första frågan två tillhörande rutor och fjärde frågan har fyra tillhörande rutor. Om en fråga har flera tillhörande rutor så kan man även ange om antalet markerade rutor är av betydelse för frågan. Exempelvis, i första frågan i figur 2.1 anges att man endast kan kryssa i en ruta, antigen man eller kvinna. I den fjärde frågan finns ingen begränsning angiven, man får kryssa i fler alternativ. Förutom detta kan man fastställa en känslighetsgrad för markerade rutor. Känslighetsgraden anges i procent och är den procentandel av rutan som är ifylld. Om man exempelvis kryssar i en ruta och sedan ändrar sig och skriver över den markerade rutan, så kan Forms känna av detta. Om rutan innehåller mer än det angivna procentvärdet, uppfattas rutan som ej ifylld. Eftersom varje ruta i ett markeringsfält inte är kopplad till sin betydelse, ger markeringar i Forms ingen information i utdatafilen. Man måste därför vid definiering av ett markeringsfält ge varje ruta ett numeriskt värde som representerar dess betydelse. Om man exempelvis i figur 2.1 kryssar i rutan *Kvinna* i första frågan kommer det inte att stå kvinna i utdatafilen. Ett exempel på hur markeringar presenteras i utdatafilen visas i avsnitt 2.2.2 Transaktionsbeskrivning.

Urvalsfält används för de fält som innehåller ord, tecken eller symboler som kan ringas in. Precis som med markeringsfält måste man tilldela urvalsfälten ett numeriskt värde när de

definieras. I figur 2.1 är den tredje frågan ett urvalsfält. Här kan man ange om endast ett alternativ får ringas in eller fler.

Bildfält används för de fält där ett svar består av ospecificerad text, det går alltså inte att definiera vilken typ av information som kommer att finnas i fältet. Detta löses genom att ange, att endast titta på fältet om det är ifyllt. Om fältet är ifyllt så får man skriva det manuellt i modulen Verify. I figur 2.1 är sista frågan ett bildfält.

Streckkodsfält är speciellt skapad för att läsa och känna av olika typer av streckkoder. Forms kan känna av de flesta streckkodstyperna, exempelvis EAN-8-streckkod som är en europeisk standard.[2]

När beskrivningen av enkätens fält och utseende är färdig sparas den i en formulärdefinition, som senare kommer att användas av Forms för att skapa en transaktionsbeskrivning.

2.2.2 Transaktionsbeskrivning

När man har en färdig formulärdefinition med definierade fält kan man vidare bestämma utseende och upplägg på utdata från ifyllda enkäter. Detta görs i en transaktionsbeskrivning, som är en mall för utdata. Det som anges i transaktionsbeskrivningen visas som resultat i utdatafilen, och här finns många möjligheter.

Transaktionsbeskrivningen skapas automatiskt av Forms när man definierar de olika fälten i en formulärdefinition, fälten som definierats läggs i den ordning man har definierat dem. Ordningen på de olika fälten kan dock modifieras för att bättre passa ett system, om man så skulle önska. Man kan dessutom välja att ta bort fält som inte ska presenteras i utdatafilen.

I transaktionsbeskrivningen kan man även bestämma att sätta in olika tecken eller symboler mellan fälten för att kunna urskilja de olika svaren i den slutgiltiga utdatafilen, fälten kan skiljas med ett tabbtecken, kommatecken eller ett annat valt tecken. Fastän man väljer att sätta in ett tecken mellan fälten tillkommer ibland nollor eller blanksteg mellan fälten, som kallas tomrum. Tomrummet uppstår om en fråga i enkäten inte besvarats, då utnyttjas inte fältets längd i utdatafilen. Därför kan man välja att komprimera utdatafilen för att ta bort outnyttjad längd.[1]

I transaktionsbeskrivningen finns en möjlighet att lägga in kommentarer, namn eller rubriker för varje fält, för att ytterligare kunna frånskilja de olika svaren eller enkäterna. En ifylld enkät från figur 2.1.1 kan till exempel presenteras i en utdatafil på följande sätt.

11	780920	31	33	44

Figur 2.2: Komprimerad utdatafil med tabbtecken som avskiljare.

11,31,33,44,00000

Figur 2.3: Utdatafil med kommatecken som avskiljning och utan komprimering.

Kön: 1 Ålder: 780920 Favoritprodukter: 1 3 Intressen: 4 Andra produkter:

Figur 2.4: Utdatafil med rubriker.

I första presentationen, figur 2.2, valdes tabbtecken för att urskilja utdatafälten och komprimering för att ta bort tomrum för fråga 5 som inte är ifylld. Här ser man att markeringsfälten och urvalsfälten valdes i formulärdefinitionen att presenteras med två siffror, där första siffran anger frågan och andra siffran anger valet. I första frågan kryssades alternativet *Man*, fråga 3 ringades *Disketter* och *Skrivare* in och i fråga 4 kryssades *Dataspel*. Det kan vara svårt att komma ihåg vad varje siffra i sådana fält betyder och därför underlättar det att göra en tabell som redovisar informationen. Se tabell 2.1. Sifferströngen på andra platsen i utdatafilen representerar ett ifyllt personnummer.

I figur 2.3 presenteras fälten med kommatecken, men ingen komprimering. Eftersom komprimering inte är vald och fråga 5 inte fylldes i visas nollor i utdatafilen. Även här presenteras markeringsfälten och urvalsfälten på samma sätt som i tidigare exempel, första siffran anger frågan och andra siffran anger valet. Ett av fälten, Fråga 2 från figur 2.1, valdes att inte presenteras här för att illustrera det som sagts tidigare att man själv kan välja vilka fält som ska redovisas i utdatafilen.

I figur 2.4 har fälten presenterats med komprimering och rubriker, därför valdes i formulärdefinitionen att endast presentera markerings – och urvalsfält med en siffra som anger svarsalternativ.

Fält	Innebörd	
Fråga 1	1 = Man 2 = Kvinna	
Fråga 3	1 = Disketter 2 = Cd-skivor	3 = Skrivare 4 = Webbkamera
Fråga 4	1 = Böcker 2 = Sport	3 = Mode 4 = Dataspel

Tabell 2.1: En tabell som redovisar fältens innebörd

2.2.3 Jobbeskrivning

När man har en färdig formulärdefinition med en tillhörande transaktionsbeskrivning är det dags att skapa en jobbeskrivning för de andra modulerna i programmet. Man skapar en jobbeskrivning genom att ange formulärdefinitionen för de enkäter som ska bearbetas av modulerna och på så sätt får modulerna instruktioner om hur enkäterna ska hanteras. Man kan välja att ange fler formulärdefinitioner i en jobbeskrivning om man ska jobba med flera olika enkäter samtidigt.

- Scanmodulen får genom jobbeskrivningen information om vilka enkäter som ska läsas in. Det är justeringsfälten och igenkänningsfälten i formulärdefinitionen som modulen då letar efter för att matcha med motsvarande fält i enkäterna.
- Interpretmodulen får veta vilka fält i enkäterna som ska tolkas och hur de ska tolkas och valideras. Exempelvis får modulen veta att frågorna 1-5 i figur 2.1 består av olika fält som ska tolkas, hur de ska tolkas och valideras anges av informationen i fältdefinitionen som beskrevs tidigare.
- Verifymodulen får instruktioner om hur den ska samverka med användaren vid den manuella korrigeringen av feltolkningar som hittas i Interpret.
- Transfermodulen får instruktioner om vilken fil den bearbetade datan ska skickas till.

2.3 Scan

När alla förberedelser har gjorts i Manager, är nästa steg att läsa in de ifyllda enkäterna. Detta görs i modulen Scan som använder en scanner för inläsning och identifiering av enkäter. Modulen kontrollerar om de inlästa enkäterna överensstämmer med justeringsfälten och/eller igenkänningsfälten från formulärdefinitionen. Om en enkät inte kan identifieras på grund av att exempelvis enkätens justeringsfält inte överensstämmer med de i formulärdefinitionen, stoppas inläsningen och ett felmeddelande visas. Därför är det viktigt att alltid arbeta med enkäternas motsvarande formulärdefinition. Men en identifiering av en enkät som misslyckas kan även ske om enkäten läses in snett eller är smutsig på något sätt.

De enkäter som identifieras sparas i programmets databas under inläsningen. När jobbet är klart och det inte finns fler enkäter att läsa in visas statistik för hur många enkäter som har lästs in, när de lästes in och hur lång tid det tog.

2.4 Interpret

Interpret tolkar och validerar informationen i de enkäter som bearbetades i Scan, för tolkning och validering följer Interpret instruktionerna som angavs i jobbeskrivningen.

Eftersom Forms använder sig av avancerade algoritmer för igenkänning av tecken, så klarar den av att tolka de flesta tecken, men det kan hända att ett svar är lite svårt att tyda. Svar som inte följer specifikationerna i formulärdefinitionen tolkas som ett fel av Interpret. Som exempelvis fråga 1 i figur 2.1, om en kund av någon anledning kryssar i bägge rutorna så tolkas det som ett fel. Om ett svar feltolkas visas inga felmeddelanden eftersom Interpret endast jobbar med att tolka och validera fälten, därför behöver användaren av programmet inte göra något här.

När Interpret har tolkat och validerat färdigt alla enkäter visas en statistik över antalet bearbetade enkäter, tiden tolkningen av enkäterna tog och antalet feltolkade tecken. Därefter skickas enkäterna med denna information för vidare arbete till modulen Verify, oavsett om de består av feltolkade fält eller inte. Men endast de som består av feltolkade fält bearbetas i Verify.

2.5 Verify

De enkäter som innehåller otolkbara tecken, ofullständig data eller andra fel som upptäcks i Interpret verifieras manuellt i Verify. Gränssnittet här är väldigt enkelt att förstå. När ett fel

ska rättas dyker en dialogruta upp som innehåller den tolkade datan och ett meddelande som beskriver felet, på skärmen visas enkäten som innehåller felet. På så sätt är det enkelt att jämföra innehållet i dialogrutan med enkäten på skärmen. Rättningen av felet skrivs i dialogrutan och ersätter då feltolkad data.

När alla fel i en enkät rättats får den statusen fullständig, vilket innebär att data i enkäten är klar att överföras till exempelvis en utdatafil. Om något svar är omöjligt för användaren att förstå, får denne själv avgöra vad som ska göras. Antingen skriver denne bara dit något eller markerar det som ej besvarat. Man kan, när alla enkäter har bearbetats, se en statistik över de verifierade enkäterna. Här visas antalet enkäter som har verifierats, hur lång tid jobbet tog och deras status.

2.6 Transfer

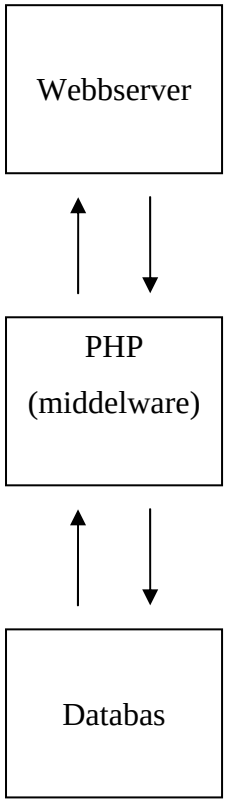
Transfer är modulen som överför informationen från de bearbetade enkäterna till en utdatafil eller ett målsystem. Utdatafilen presenterar informationen enligt de layoutregler som angavs i transaktionsbeskrivningen. Endast de enkäter med statusen fullständig från modulen Verify kommer att överföras i denna modul.

3 Prifloat

För att nå målet att formatera utdatafilen som skapas i Forms till ett format som passar in i Prifloat Infosystems databas, behöver vi veta vilket format databasen kan hantera. Avsnitt 3.1 ger en kort beskrivning av Prifloat Infosystem som innefattar ett webbaserat verktyg och Prifloat Infosystems databas. Det webbaserade Prifloatverktyget används vid olika undersökningar, exempelvis ett företag som vill undersöka sin personals trivsel, och svaren från undersökningarna lagras i databasen. Processen att skapa enkätundersökningar och lagra svaren i databasen tar idag lång tid att utföra på grund av att det sistnämnda sker manuellt. Avsnitt 3.2 presenterar en mall som vi senare i kapitel 4 ska följa när vi formaterar utdatafilen. Mallen följer det format som databasen kan hantera. Detta kapitel ska ge läsaren en överblick av vad Prifloat Infosystem är och presentera mallen som vi senare i kapitel 4 ska följa när vi formaterar utdatafilen.

3.1 Prifloat Infosystem

Prifloat Infosystem är ett undersökningssystem som innefattar ett webbaserat verktyg och en MySQLdatabas för att göra enkätundersökningar. Det webbaserade verktyget är byggt i en Linuxmiljö med Apache som webbserver. Webbservern använder PHP som middelware då den gör förfrågningar mot databasen, se figur 3.1. Figuren visar hur PHP möjliggör en anslutning från webbservern till databasen.



Figur 3.1: Visar hur databas och webbserver interagerar

Prifloatverktyget, som är ett centraliserat verktyg, används av olika företag och myndigheter för olika typer av analyser och undersökningar. Med Prifloat kan företag vid sina utvecklingsarbeten samla information om sina kunder, medarbetare och leverantörer. Detta kan göras genom exempelvis en e-postenkät, där företaget skickar e-postenkäter till sina kunder för en specifik undersökning [3]. Även andra användningsområden finns. Den som använder verktyget kan forma undersökningar efter önskat ändamål. Olika typer av frågor finns redan färdiga men det går att modifiera dem på olika sätt och anpassa efter egna behov.

För att få använda Prifloatverktyget måste man vara registrerad som användare. På Prifloats hemsida [3] kan man då logga in och börja använda verktyget. Svaren från enkätundersökningarna lagras i den tillhörande MySQL relationsdatabasen, där svaren sedan analyseras genom olika urval av de svar som samlats in. Exempelvis av alla som köpte en bil under mars månad år 2003, hur många var det som köpte en BMW.

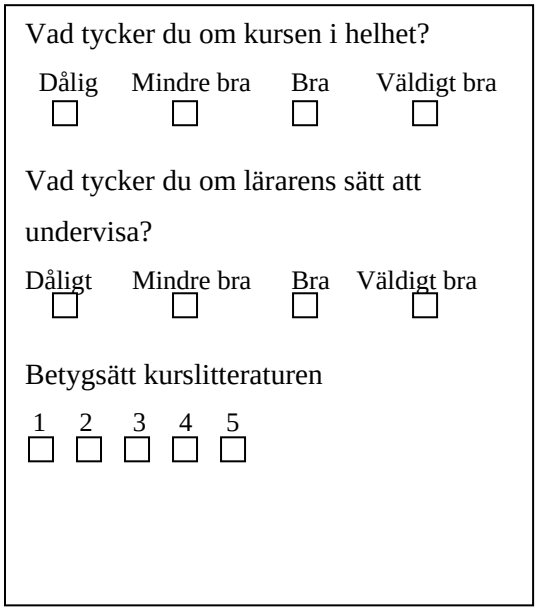
3.2 Mallen

Vi har fått en mall, av uppdragsgivaren, att följa när vi ska analysera och anpassa utdatafilen till ett format som passar i databasen. Eftersom databasen är stor och komplex skulle det ta för lång tid för oss att själva hitta en bra mall. Mallen i figur 3.2 visar vilket format utdatafilen ska ha för att lagras i databasen, det är alltså den typen av frågor och svar som mappas i databasen.

Figur 3.2: Utseendemall för indata

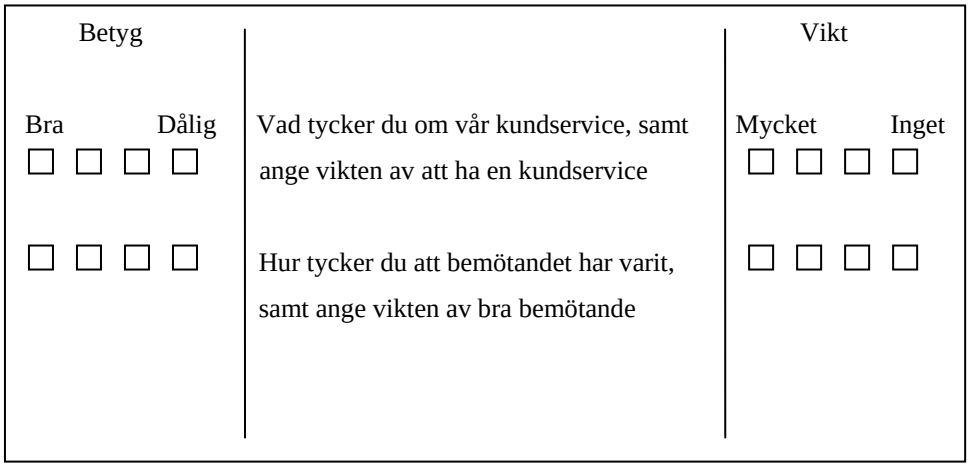
Figur 3.2 visar alla de typer av frågor som Prifloatverktyget använder vid analyser och undersökningar. Mallen visar även hur svaren från frågorna ska se ut för att sedan kunna läggas in i databasen. Här kan man se att Prifloat har sju olika typer av frågor i sina enkäter, dessa är namn, email, kategori, betyg, alternativ, index och text.

Svaren från typerna namn, email och text kommer att bestå av text i olika former så som figuren visar. Namn och email består av namnet och email - adressen på den som svarar på enkäten och text består av kommentarer. I figur 3.2 på sista raden kan man se att personen i fråga har en kommentar, *super bra!* Svaren från typerna kategori, betyg, alternativ och index kommer att bestå av siffror som anger de givna svarsalternativen. Typen kategori används för frågor som klassificerar den person som svarar på enkäten, exempelvis frågor som ”vad har du för bil”, ”vilken avdelning jobbar du på”, ”vilket universitet studerar du vid” och så vidare. Man kan endast välja ett svarsalternativ. Betyg används för frågor där man ska betygsätta något, se figur 3.3. Även här kan man endast välja ett svarsalternativ.



Figur 3.3: En enkät med frågor av typen betyg

Typen alternativ används för frågor där man kan ange flera svarsalternativ. Det kan vara frågor som ”Vilka är dina intressen”, ”Vilka städer har du bott i?” och så vidare. Typen index används för frågor där man betygsätter något och anger dess vikt, se frågorna i figur 3.4.



Figur 3.4: En enkät med frågor av typen index

4 Analys och anpassning av utdatafiler

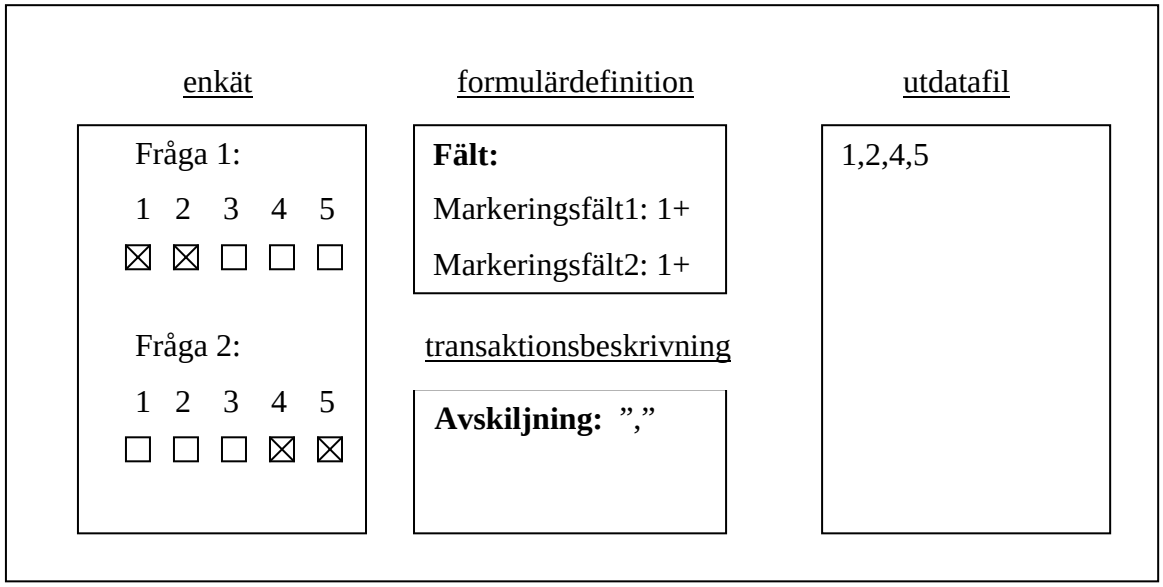
Hittills har vi använt Readsoft Forms i syfte att förstå hur den automatiska inläsningen och bearbetningen av enkäter går till och att lära oss hur utdatafiler skapas. Vi har med hjälp av modulen Manager skapat formulärdefinitioner, transaktionsbeskrivningar och jobbeskrivningar för att förbereda de andra modulerna inför inläsningen och bearbetningen samt för att skapa utdatafiler. I detta kapitel kommer vi att, utifrån vår erfarenhet av Forms, forma utdatafilen så att den ser ut som mallen i kapitel 3. Vi börjar med att följa mallen enbart genom att använda Forms, i avsnitt 4.1 beskriver och testar vi två möjliga sätt att presentera utdatafiler på i Forms. Detta ska ge läsaren en förståelse för varför vi sedan i avsnitt 4.2 bestämmer att gå vidare och ta till andra hjälpmedel. I avsnitt 4.2 beskriver vi PHP, varför vi har valt att använda PHP och hur vi med PHP har skapat en utdatafil som följer mallen i kapitel 3.

4.1 Forms utdatafil

I kapitel 2 visade vi att när man definierar markeringsfält och urvalsfält i formulärdefinitionen måste man alltid tilldela fälten numeriska värden som representerar svarens betydelse i utdatafilen. Vidare visade vi en tabell, tabell 2.1, för att kunna hålla reda på vad de numeriska värdena representerar. Man kan bestämma att låta en siffra i ett numeriskt värde representera frågan eller i transaktionsbeskrivningen sätta rubrik på frågan. När det gäller att forma utseendet på en utdatafil finns det många sätt att göra det på, och i vårt fall måste vi hitta ett sätt som följer mallen i kapitel 3.

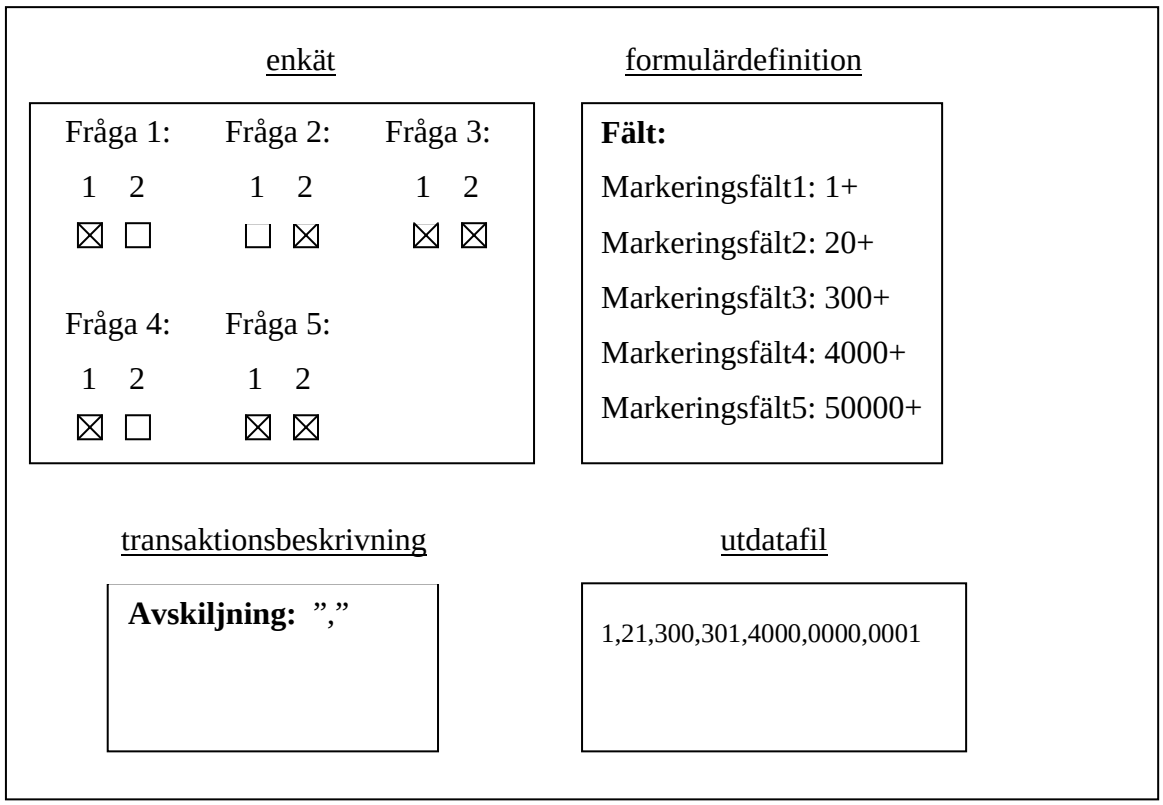
4.1.1 Två alternativ att presentera utdatafil på

Figur 4.1 visar ett exempel på hur svaren från en enkel enkät kan se ut efter att man har skapat en formulärdefinition och en transaktionsbeskrivning, frågorna är av typen markeringsfält som består av fem möjliga svarsalternativ. Figuren visar en abstraktion av formulärdefinitionen och transaktionsbeskrivningen för att simplifiera exemplet, i verkligheten är de mer komplicerade än så. I formulärdefinitionen tilldelas varje fält värdet ”1+” vilket innebär att varje ruta får ett värde i stigande ordning med 1 som startvärde. I transaktionsbeskrivningen väljs ett kommatecken som avskiljare mellan svaren. Resultatet visas i utdatafilen i figuren och här ser man att alternativen 1, 2, 4 och 5 kryssades för, men det ger inte någon begriplig information. Därför måste vi modifiera utdatafilen ännu mer.



Figur 4.1: Exempel på en enkät, en formulärdefinition, en transaktionsbeskrivning och en utdatafil.

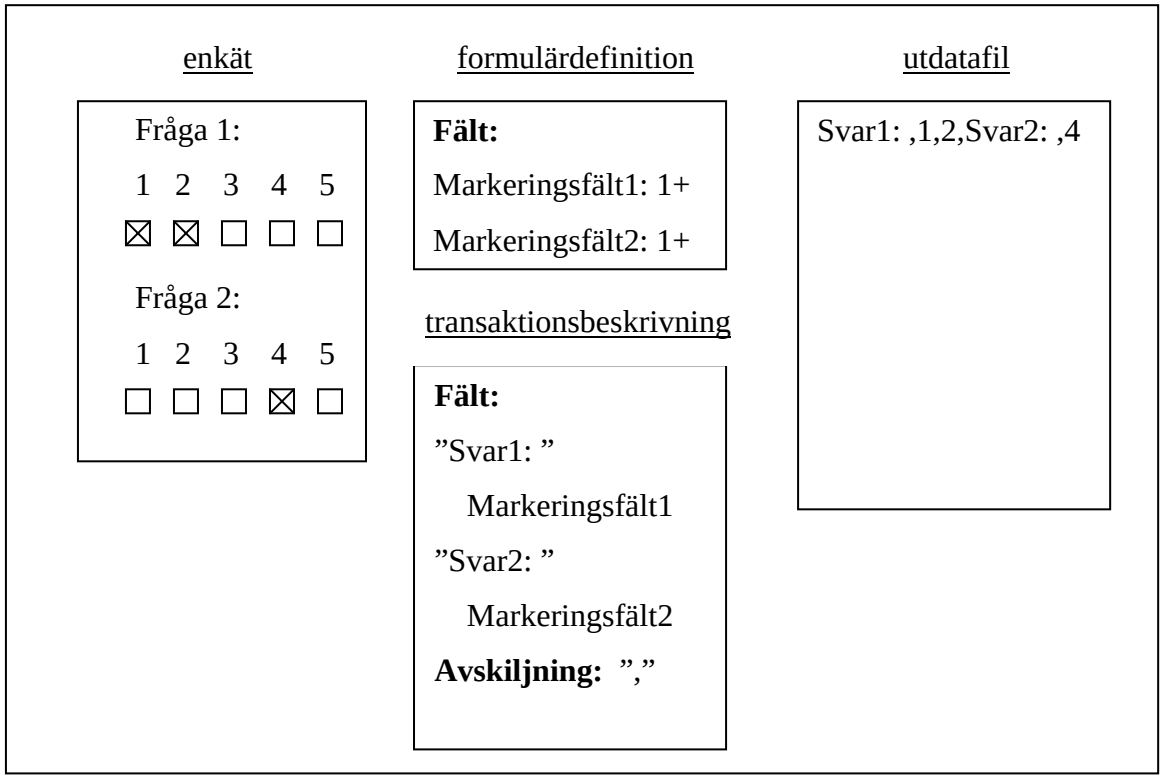
För att få en mer förståelig utdatafil finns två möjliga alternativ, det ena är att tilldela varje fråga ett unikt värde och det andra är att använda rubriker framför svaren. Vi testar det första alternativet, att tilldela varje fråga ett unikt värde, genom att kontrollera hur stora värden vi kan skriva i transaktionsbeskrivningen. Detta är viktigt eftersom vissa frågor kan ha många svarsalternativ och vi måste veta vad Forms har för begränsningar. I figur 4.2 visas ett exempel på hur vi har utfört testet. Fråga 1 får värdet 1+, fråga 2 får värdet 20+, fråga 3 får värdet 300+, fråga 4 får 4000+ och fråga 5 får 50000+. Det räcker med att testa fem frågor, även om det verkar lite, men det kan knappast vara möjligt att en enkät består av 10000 svarsalternativ.



Figur 4.2: Exempel på en enkät med unikt definierade värden.

Som man ser i exemplet är det något som inte stämmer med svaren i fråga 5, den första siffran kom inte med i utdatafilen. Slutsatsen som dras från detta är att man kan värdesätta svar från 1- 9999.

Det andra alternativet, att lägga rubrik på en fråga testar vi också. Figur 4.3 visar ett exempel på hur det kan se ut. Som figuren visar måste man fortfarande värdesätta dessa fält i formulärdefinitionen, men eftersom vi kommer att använda oss av rubriker går det bra att använda samma värde, 1+. Resultatet är nu en tydligare utdatafil, som mer tydligt visar att svarsalternativen ett och två i första frågan är ifyllda och svarsalternativet fyra är ifyllt i andra frågan. Detta alternativ är ett väldigt enkelt sätt att följa mallen från kapitel 3. Men nackdelen är att när man skriver rubriker för varje fråga i en enkät utför man ett väldigt tidskrävande arbete, eftersom det ska göras i transaktionsbeskrivningen. Dels ska man först i formulärdefinitionen värdesätta varje svar, vilket är ett måste för markeringsfält och urvalsfält. Och dels måste man senare även jobba lite extra i transaktionsbeskrivningen för att sätta rubriker. I det tidigare alternativet behövde man endast sätta unika värden i formulärdefinitionen för att få ett tydligt svar.



Figur 4.3: Exempel på en enkät, en formulärdefinition, en transaktionsbeskrivning och en utdatafil.

4.1.2 Analys av de två alternativen

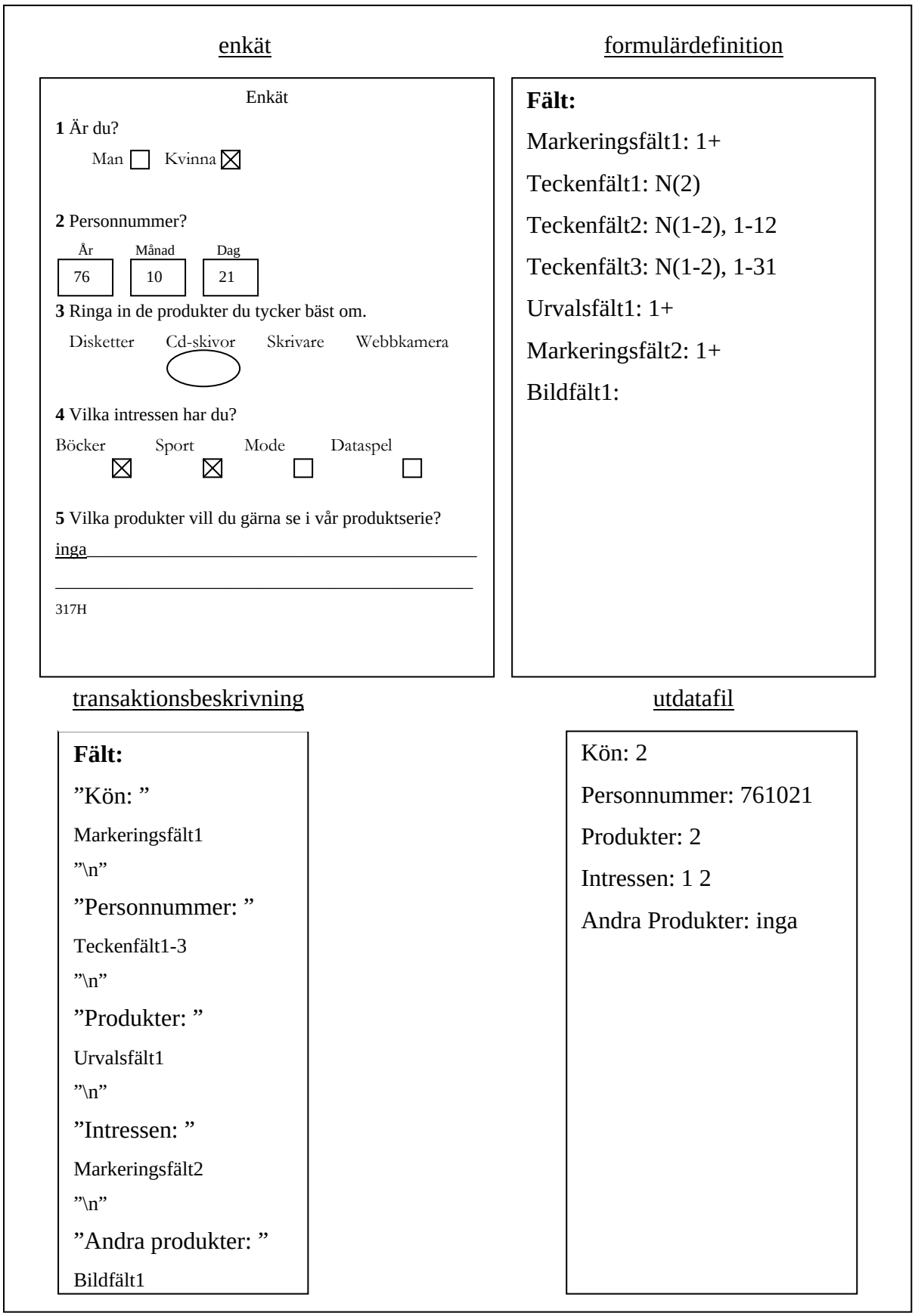
Alternativen som beskrivs ovan ger ingen bra lösning. I det första alternativet är det Forms som sätter gränser. Det andra alternativet kräver mycket arbete av användaren, särskilt om arbetet ska utföras för en enkät som består av väldigt många frågor. Med det andra alternativet lyckades vi ändå till en viss del följa mallen i kapitel 3, där varje svar föregås av en rubrik. Därför tänker vi fortsätta med det andra alternativet och försöka följa mallen fullkomligt.

Som figurerna 4.1 – 4.3 visar, presenteras svaren i utdatafilen i en sekvens, det beror på en defaultkonfiguration och för att fullkomligt kunna följa mallen i kapitel 3 måste varje svarsfält presenteras radvis. Det åstadkommer man på samma sätt som med rubrikerna i transaktionsbeskrivningen, men istället för att skriva en rubrik skriver man ”\n” som sätts efter varje svar för att bryta den sekventiella raden. Forms tolkar det som ett kommando för att skifta resterande delen av sekvensen till en ny rad. Figur 4.4 visar enkäten från figur 2.1 i kapitel 2 och hur vi i transaktionsbeskrivningen gjorde för att få vår utdatafil att se ut som mallen i kapitel 3.

Utdatafilen presenteras med rubriker före varje svar och varje svarsfält kommer på en ny rad, precis med samma struktur som mallen i kapitel 3. I formulärdefinitionen har vi angett de formatspecifikationer för teckenfälten som vi beskrev i avsnitt 2.2.1. Som angivet i figuren får rutan *År* i andra frågan anta ett tvåsiffrigt värde, och de andra rutorna får anta ett eller tvåsiffriga värden. För de andra rutorna har vi även angett vilka värden som är acceptabla, 1-12 för rutan *Månad* och 1-31 för rutan *Dag*.

Som figur 4.4 visar ger det andra alternativet med rubriker och ”\n” en förståelig utdatafil som följer mallen i kapitel 3. Men som tidigare sagts är alternativet tidskrävande och med ”\n” efter svarsfälten tar det ännu längre tid att utföra arbetet.

Med denna information, att Forms begränsar värdesättning av markerings – och urvalsfält och att det är tidskrävande att sätta rubriker framför svarsfälten, ska vi nu gå vidare och försöka lösa problemet med andra hjälpmedel.



Figur 4.4: Exempel på vad som ska göras för att få utdatafilen att följa mallen i kapitel 3.

4.2 PHP

Hypertext Preprocessor, PHP [4], är ett skriptspråk. PHP bäddas in i, Hyper Text Markup Language, HTML. PHP genererar dynamisk kod till skillnad från endast ren HTML som ger statisk kod. HTML-kod skrivs ner i en fil, filen laddas sedan upp till en webbplats och visas som en webbsida. HTML-koden är oförändrad tills nästa gång man laddar ner filen för att uppdatera dess innehåll. Det är det som gör att ren HTML är statiskt.

Webbsidor som innehåller PHP-kod lagras i en server och jobbar oftast mot databaser. Om datan ändras i databasen behöver man inte ändra koden för webbsidan, eftersom PHP-koden hämtar information från databasen varje gång man öppnar sidan. Det är det som gör att PHP är dynamisk. [5]

Det spelar ingen roll vilket operativsystem servern har eftersom PHP är ett plattformsoberoende språk, fördelen med det är att om man vill flytta PHP från ett operativsystem till ett annat så krävs det inga ändringar, förutsatt att man inte använt sig av systemfunktioner för ett specifikt operativsystem i koden.

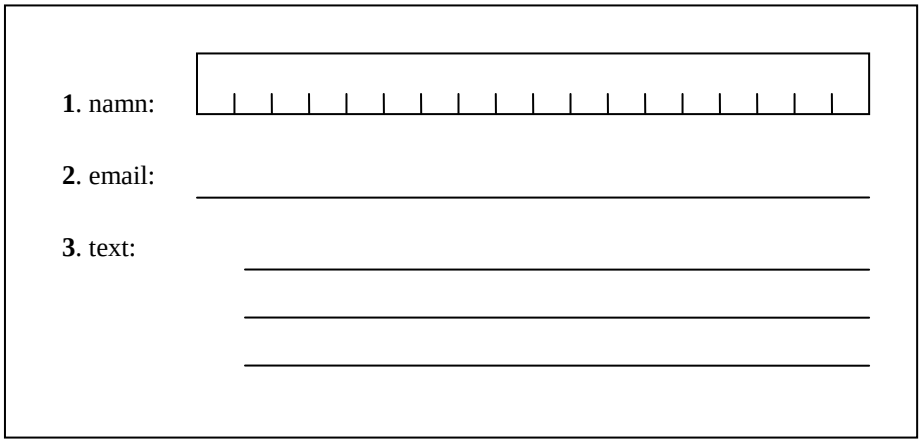
4.2.1 Varför PHP?

Vi såg i avsnitt 4.1 att det var möjligt att skapa utdatafiler i Forms som följer mallen i kapitel 3. Men det var tidskrävande. Det vore enklare att bestämma sig för ett sätt att definiera varje typ av fråga i Forms, för alla enkäter. Exempelvis att definiera alla betygsfrågor på ett sätt, alla indexfrågor på ett annat sätt tills varje typ av fråga har definierats. Definieringen ska fungera på så vis att det krävs så lite arbete i transaktionsbeskrivningen som möjligt. Exempelvis så kan man definiera alla betygsfrågor med ett värde X, värdet kan återanvändas för alla betygsfrågor och representera svaret i utdatafilen. Utdatafilen kommer att visa värdet X för alla betygsfrågor.

Det är sedan möjligt att skapa en programmodul som läser av utdatafilen och genererar en ny fil som vi kallar ”dataut”. Den nya filen innehåller svaren från utdatafilen formaterade enligt mallen som introducerades i kapitel 3. Exempelvis, varje gång programmodulen läser av värdet X i utdatafilen från Forms läggs informationen ”betyg: (svar)” i filen ”dataut”.
Vi valde att skapa programmodulen i PHP eftersom data läggs in i databasen med hjälp av ett PHP gränssnitt. PHP är också enkelt att komma igång med och använda tack vare webbsidan http://www.php.net som är en komplett guide över alla funktioner och verktyg som PHP har att erbjuda.

4.2.2 Vilken definition har bestämts

En enkel enkät kommer att användas som exempel för att visa hur de olika typer av frågor ska definieras, för att användas i samband men programmodulen som beskrevs i föregående avsnitt. Enkäten kommer att utökas stegvis för varje typ av fråga. Figur 4.5 visar en enkät med tre frågor där svaren består av text.



Figur 4.5: Enkät med textfrågor.

I utdatafilen kommer svaren från dessa tre frågor att sitta ihop till en enda textsträng. Som vi tidigare såg var det möjligt att skilja på de olika svaren med hjälp av ett tecken, vi valde då att använda kommatecken som avskiljare. Men att använda sig av kommatecken som avskiljare är olämpligt eftersom det är ett tecken som kan anges när man svarar på den tredje frågan i figur 4.5. Vi bestämmer oss för att använda tecknet # som avskiljare, för att det är ett av de få tecken som kan användas i svar. Om den som svarar på enkäten i figur 4.5 ändå skriver in tecknet # i svaret kan problem uppstå, men det är lätt att åtgärda i Forms. Fråga 3 i figuren definieras som ett bildfält i Forms och som vi tidigare har sagt i kapitel 2 skrivs svaren för ett bildfält manuellt in i Forms. Användaren av programmet kan då bestämma att ta bort # för att åtgärda problemet.

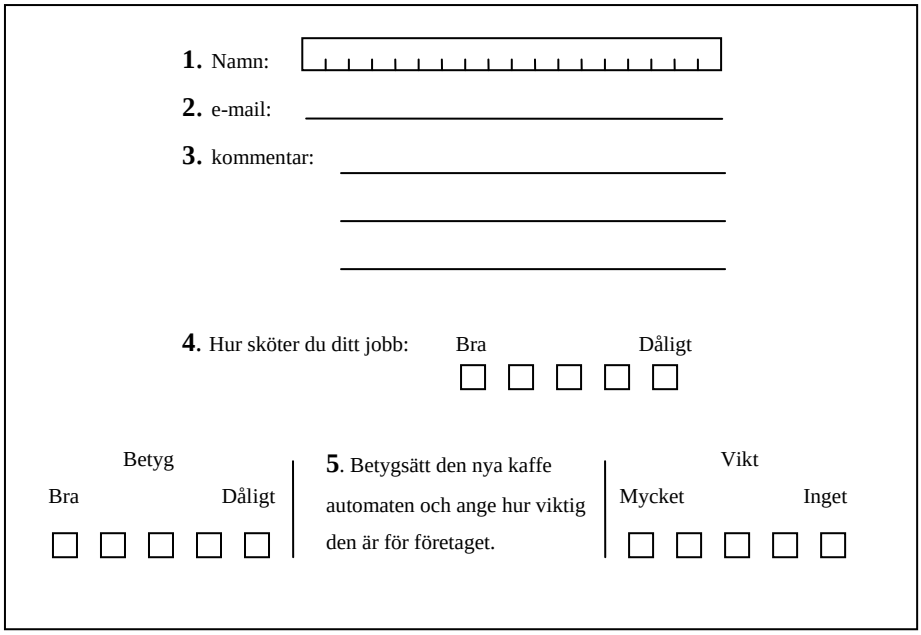
För att programmodulen ska känna igen och hantera svaren, i utdatafilen, från frågorna i figur 4.5 måste de definieras på ett lämpligt sätt. Att lägga till rubriker i transaktionsbeskrivningen är det som passar bäst för dessa tre typer av frågor, eftersom enbart svaren från dessa kommer att visas i utdatafilen. För namnfrågor läggs rubriken ”namn”, för emailfrågor läggs rubriken ”email” in och för fri text läggs rubriken ”text”. Om man exempelvis svarar på frågorna i figur 4.5 på följande sätt:

1.namn: Andreas Nilsson, 2.email: nilsandr@mail.com, 3.text: Inga kommentarer

kommer svaren att få följande utseende i utdatafilen:

namn#Andreas Nilsson#email#nilsandr@mail.com#text#Inga kommentarer.

Enkäten i figur 4.5 får ytterligare två frågor och det framgår i figur 4.6, där fråga 4 är en betygfråga och fråga 5 är en indexfråga.



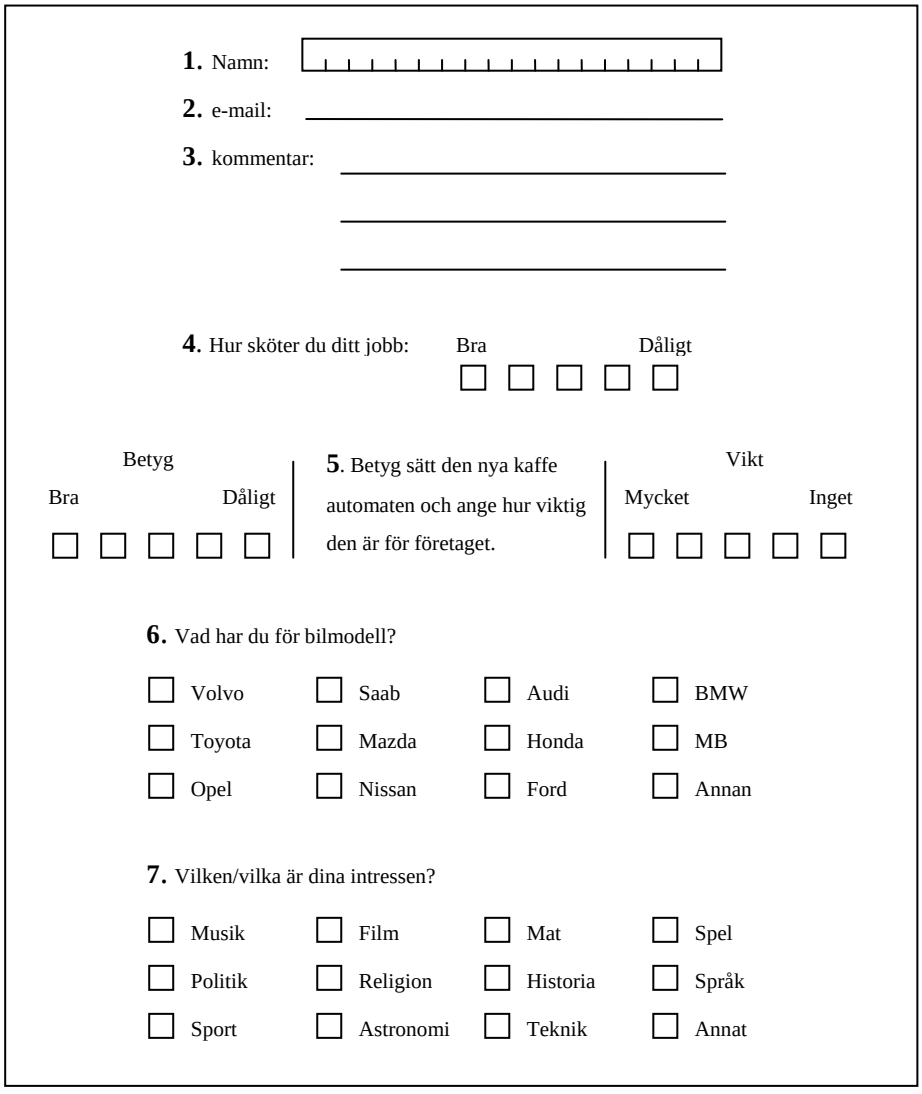
Figur 4.6: Betyg - och indexfrågor läggs in.

Både betyg - och indexfrågor fungerar på så sätt att man kan markera in sitt svar. Därav definieras dessa frågor som markeringsfrågor i Forms och som vi visade i kapitel 2 måste dessa tilldelas numeriska värden i formulärdefinitionen. I avsnitt 4.1 förklarade vi nackdelen med att använda rubriker för markeringsfrågor, det är tidskrävande eftersom man utför ett jobb både i formulärdefinitionen och i transaktionsbeskrivningen. På grund av detta kommer vi inte att använda rubriker för betyg – och indexfrågor utan endast tilldela dessa numeriska värden. Vi väljer att definiera alla betygfrågor med värdet 9981+ och indexfrågor med värdet 9991+.

Värdena 9981 – 9989 i utdatafilen representerar en betygfråga och dess svarsalternativ. De tre första siffrorna representerar frågetypen och den sista representerar svarsalternativet. I kommande kapitel, kapitel 5, visas hur dessa värden kommer att hanteras av vår programmodul. Som figur 4.6 visar har betygfrågan en betygskala och sannolikheten att en betygskala, i de typer av enkäter som Prifloat använder, värderas med fler än 9 svarsalternativ är liten.

I figur 4.6 visas att en indexfråga är uppdelad i två delar, en betygdel och en viktdel. Värdena 9991 – 9999 representerar en indexfråga och svaret för betygdelen, där de tre första siffrorna representerar frågetypen och den sista siffran representerar betygdelen. För viktdelen spelar det ingen roll vilka värden vi väljer vid definieringen, eftersom viktdelen ska sättas tillsammans med betygdelen till ett svar, se figur 3.2 i kapitel 3. Hur detta ska ske kommer att förklaras i kapitel 5. Vi väljer att definiera viktdelen endast med det värdet som ska representera svaret, 1+.

Enkäten får ytterligare två frågor, fråga 6 som är en kategorifråga och fråga 7 som är en alternativfråga, se figur 4.7. Dessa frågor definieras, precis som betyg - och indexfrågor, som markeringsfrågor i Forms.



Figur 4.7: Kategori och alternativ frågor läggs in.

Vi väljer att definiera kategorifrågor med värdet 9701+, eftersom kategorifrågor är typer av frågor som kan ha många svarsalternativ till skillnad från betyg – och indexfrågor som beskrevs ovan. Värdena 9701 – 9799 representerar en kategorifråga och dess svarsalternativ, där de två första siffrorna representerar frågetypen och de två sista representerar svarsalternativet. Detta visar att en kategorifråga kan ha 99 svarsalternativ.

Fråga 7 i figur 4.7 är en alternativfråga och som figuren visar får man markera fler än ett svarsalternativ. På grund av detta kan problem uppstå när man definierar en alternativfråga med ett numeriskt värde så som vi har definierat betyg -, index – och kategorifrågor. Exempelvis, anta att vi definierar alternativfrågor med värdet 8801+, och att en person har besvarat enkäten i figur 4.7. Personen i fråga har kryssat för både Musik och Mat som intressen i fråga 7, det vill säga svarsalternativen 1 och 7. I utdatafilen kommer svaren från fråga 7 att se ut på följande sätt: 8801#8807.

Detta beror på att Forms lägger hela det numeriska värdet från ett markerat svar i utdatafilen. Det blir svårt för programmodulen att avgöra om svaren tillhör en och samma alternativfråga eller om det är två frågor som har besvarats. Därför väljer vi att använda en rubrik för alternativfrågor, rubriken som vi väljer att definiera alternativfrågor med är ”alternativ”. Men vi måste fortfarande tilldela alternativfrågor ett numeriskt värde eftersom dessa är markeringsfrågor. Det numeriska värdet som vi väljer att tilldela alternativfrågor är 1+, som representerar svarsalternativet. Med dessa val som vi gör kan man med exemplet ovan (en person kryssar i svarsalternativen Musik och Mat i fråga 7 i figur 4.7) se att utdatafilen får följande utseende: alternativ#1#7

Nu blir det lättare för programmodulen att avgöra om svaren tillhör en och samma fråga eller två skilda. Det finns dock en nackdel med att ange rubriker för markeringsfrågor, som vi tidigare i avsnitt 4.1 visade krävs att man utför ett arbete i formulärdefinitionen och i transaktionsbeskrivningen. Men detta är nödvändigt för att få programmodulen att hantera alternativfrågor på ett korrekt sätt.

Sättet vi har valt att definiera de olika typer av frågor har inga begränsningar gällande antalet frågor som en enkät får har. Det gör att programmodulen klarar av alla enkäter som har frågetyperna namn, email, text, betyg, index, kategori och alternativ.

Tabell 4.1 visar vilka värden och rubriker vi har valt att sätta för alla typer av frågor som vi har visat i detta kapitel. Utdatafilen från Forms ska presentera data med de angivna värden och rubrikerna för att vår programmodul ska kunna formatera den enligt mallen i kapitel 3.

Frågetyp	Värde	Rubrik
Namn		namn
Email		email
Text		text
Kategori	9701+	
Index	9991+	
Betyg	9981+	
Alternativ	1+	alternativ

Tabell 4.1: Visar vad vi har valt att tilldela varje frågetyp.

5 Implementering

För att formatera utdatafilen enligt mallen i kapitel 3 skrev vi en modul i PHP, nedan följer en redogörelse för vad modulen gör.

namn#Anna Andersson#email#anna@mail.com#text##9982#9992#1#9707#alternativ#1
#4#11
namn#Bertil Ekdal#email#bertil@mail.com#text# #9981#9994#2#9710#alternativ#3

Figur 5.1: Utdatafil som följer vår tabell i kapitel 4.

Figur 5.1 visar en utdatafil där datan presenteras enligt tabell 4.1 och svaren kommer från två ifyllda enkäter i figur 4.7. För att enskilt kunna hantera varje värde, rubrik och svarsfält i utdatafilen vill vi sätta dessa på var sin rad. Detta gör vi med hjälp av den inbyggda funktionen `split(pattern, string)`. Funktionen returnerar datan (string) i en array av strängar, där varje sträng som föregås av mönstret (pattern) börjar på en ny rad.

```
split(”#”, $filedata);
```

Med denna funktion kommer datan från utdatafilen att få följande utseende, se figur 5.2.

0	namn
1	Anna Andersson
2	email
3	anna@mail.com
4	text
5	
6	9982
7	9992
8	1
9	9707
10	alternativ
11	1
12	4
13	11
14	
16	namn
17	Bertil Ekdal
	osv.

Figur 5.2: Arrayen `delar[]`, efter att funktionen `split()` utförts på utdatafilen

Nu när datan är uppdelad ska modulen gå igenom arrayen rad för rad, det görs med hjälp av en while loop och en variabel `P` som håller reda på aktuell position. Första strängen i arrayen `delar[]` är rubriken ”namn”, och finns på position 0 i arrayen. När ”namn” påträffas sätts ”namn”, ett kolon ”:” och svaret (svaret är Bertil Bertilsson i figur 5.2) ihop i en ny array `nya[]`, se figur 5.3. Sen ökar variabeln `P` till position 2, som är nästa sträng som ska läsas av.

```
$nya[] = $delar[$P] . ”: ” . $delar[$P+1];
$P += 2;
```

På position 2 i figur 5.2 finns rubriken ”email” och den bearbetas på exakt samma sätt som rubriken ”namn”, där rubriken, ett kolon och svaret sätts ihop i den nya arrayen `nya[]`. Sen ökar variabeln `P` till position 4. Samma princip gäller för rubriken ”text”. Men här sätts en variabel som representerar rubrikens indexsiffra.

```
$nya[] = $delar[$P] . ++$text . ”: ” . $delar[$P+1];
```

```
$P += 2;
```

När en array i PHP tilldelas värden så som arrayen `nya[]` får värden tilldelat, behöver man inte ange på vilken position värdet ska hamna. PHP ökar automatiskt tilldelningspositionen för arrayen `nya[]`. Därför kommer varje rubrik, kolon och svar att läggas i rätt position i arrayen `nya[]`, se figur 5.3. Figuren visar en utskrift av arrayen `nya[]`, där varje rad i utskriften motsvarar en position i arrayen.

Nästa sträng, på position 6, är ett numeriskt värde 9982 där 998 representerar en betygsfråga och 2 är det valda svarsalternativet. På tredje positionen i strängen sätts ett kolon ”: ” in, det vill säga efter värdet 998. När värdet 998 påträffas ersätts det med rubriken ”betyg” och precis som med frågetypen ”text” sätts en variabel här `$betyg` som representerar rubrikens indexsiffra. Rubriken, indexsiffran, kolonet och svarsalternativet sätts sedan in i den nya arrayen `nya[]`. Därefter ökas variabeln `P` med ett steg till position 7 i arrayen.

```
$delar[$P] = substr(chunk_split($delar[$P], 3, ’: ’), 0, 6);
$delar[$P] = substr_replace($delar[$P], ”betyg”. ++$betyg, 0, 3);
$nya[] = $delar[$P];
$P++;
```

På position 7 finns värdet 9992, där 999 representerar frågetypen och 2 svarsalternativet. På tredje positionen i strängen sätts ett kolon ”: ” in, det vill säga efter värdet 999. Värdet 999 ersätts med rubriken ”index” och efter sätts en indexsiffra `$index`. Rubriken, indexsiffran, kolonet och svarsalternativet sätts ihop med strängen på position 8, med ett bindestreck emellan. Resultatet placeras i den nya arrayen `nya[]`, se figur 5.3. Variabeln `P` ökas här med två steg till position 9.

```
$delar[$P] = substr(chunk_split($delar[$P], 3, ’: ’), 0, 6);
$delar[$P] = substr_replace($delar[$P], ”index”. ++$index, 0, 3);
$nya[] = $delar[$P] . ”-” . $delar[$P+1];
$P += 2;
```

På position 9 finns värdet 9707. De första två siffrorna, 97, representerar frågetypen kategori och ersätts därför med rubriken ”kategori”. Rubriken efterföljs av en indexsiffra, ett

kolon och svarsalternativet. Variabeln `P` ökas med ett steg till position 10. Frågetypen ”kategori” följer nästan samma struktur som ”betyg”, skillnaden är värdet som ersätts, eftersom varje frågetyp representeras av unikt värde. Därför visas inte koden för ”kategori” med här.

På position 10 finns rubriken ”alternativ”. När rubriken ”alternativ” påträffas sätts en indexsiffra och ett kolon efter den och därefter letar modulen efter de valda svarsalternativen i de efterföljande positionerna. Så länge som svaren i efterföljande positioner har ett värde som är mindre än 1000 uppfattas dessa som ett tillhörande svar. De tillhörande svaren sätts ihop med ett bindestreck emellan, precis på samma sätt som i typen ”index”. Därefter sätts svarsalternativen ihop med rubriken och placeras i den nya arrayen `nya[]`, se figur 5.3.

```
-----Svar 1-----
namn: Anna Andersson
email: anna@mail.com
text1:
betyg1: 2
index1: 2-1
kategori1: 7
alternativ1: 1-4-11
-----Svar 2-----
namn: Bertil Ekdal
email: bertil@mail.com
text1:
betyg1: 1
index1: 4-2
kategori1: 10
alternativ1: 3
```

Figur 5.3: Utseendet på den nya arrayen `nya[]`.

Som figuren 5.1 visar börjar varje enskild enkät på en ny rad, och varje gång en ny rad påträffas nollställs alla variabler och en rubrik för den nya enkäten sätts, se figur 5.3.

6 Resultat och rekommendationer

Huvudmålet med projektet var att skapa en enkätkonverterare som kan transformera Readsoft Forms utdatafiler till ett format som passar Prifloat Infosystems databas. Detta mål har vi nått.

6.1 Resultat

Nedan mäter vi de delmål som sattes upp för att nå huvudmålet. Därefter följer rekommendationer menat till dem som ska göra ett liknande arbete.

6.1.1 Studera Forms

För att nå huvudmålet var vi tvungna att börja jobba med programvaran Readsoft Forms. Med Forms kunde vi automatiskt läsa in enkäter med en scanner och sedan presentera informationen från enkäterna i en utdatafil. Inläsningen av enkäterna utfördes med en scanner och det var inga problem att använda den. Däremot var det tidskrävande att skapa utdatafiler, på grund av att det finns så många olika sätt att presentera en utdatafil på. Målet med detta delmål var att lära oss använda programvaran för att automatiskt läsa in handskrivna enkäter och skapa utdatafiler. Vi använde oss av Readsoft Forms handbok [1] som steg för steg visade hur vi kunde använda programmets fem moduler för att nå delmålet. I och med att vi lärde oss att hantera programmet och skapa utdatafiler som Forms ger nådde vi delmålet.

6.1.2 Studera Prifloat Infosystems databas

För att hitta ett format på utdatafilen som passar in i databasen skulle vi förstå databasens uppbyggnad. Men den var alldeles för stor och komplex för att vi skulle hinna göra det och därför fick vi en mall av uppdragsgivaren, se avsnitt 3.2. Mallen följer, enligt uppdragsgivaren, ett format som passar in i databasen.

6.1.3 Analysera och anpassa utdatafiler

För att kunna formatera utdatafilen efter mallen som presenterades i kapitel 3 gjorde vi en analys av utdatafilerna i Forms och beskrev olika sätt som man kan skapa en utdatafil på. Analysdelen av rapporten var väldigt arbetskrävande eftersom det finns så många olika sätt att skapa en utdatafil i Forms. Vår uppgift var att hitta en utdatafil som följer mallen i kapitel 3. Med enbart Forms visade det sig vara tidskrävande att försöka lösa det och även resultatlöst på grund av Forms begränsningar. Vi gick därför vidare med att ta hjälp av PHP. Med PHP skrev vi ett program som formaterar en utdatafil till mallens format. Vi skapade utdatafilen i

Forms där frågetyperna tilldelades värden och rubriker enligt tabell 4.1. Denna tabell ska underlätta arbetet i Forms, den som använder programvaran ska endast följa tabellen vid skapandet av utdatafiler för att få programmodulen att formatera utdatafilen. Eftersom vi tidigare hade fördjupat oss i Forms när vi gjorde analysdelen och hade även tidigare använt PHP var det inga svårigheter för oss att anpassa och formatera utdatafilen enligt mallen i kapitel 3.

6.1.4 Lagra utdatafil i databasen

Detta delmål hann vi inte med, analysdelen tog längre tid än vi hade räknat med. Det var dock inget hinder för att nå huvudmålet eftersom detta delmål inte hörde till huvudmålet och skulle endast göras i mån av tid.

6.2 Rekommendationer

För de personer som ska göra ett liknande arbete rekommenderar vi att tänka igenom det ni ska göra och skriva en väl planerad tidsplan. Se också till att alltid skriva ner det ni gör och få med de problem som ni stöter på. Detta är väldigt viktigt för rapporten.

När man jobbar med ett nytt ämne, som detta projekt var för oss i början, är det viktigt att vara i det klara med vad som ska uppnås. Annars kan det bli ett hinder för det fortsatta arbetet, vilket hände i vårt fall. Vi studerade programvaran Forms utan att riktigt veta hur pass omfattande nästa steg skulle bli, där vi skulle studera databasen. Vi hade ingen aning om databasens komplexitet och fick därför ändra synvinkel på kapitel 3. Därför är det bra att i början alltid vara säker på vad som ska göras och dessutom ligga på samma nivå som uppdragsgivaren och handledaren.

7 Summering av projektet

När vi började arbeta med projektet visste vi inte hur pass svårt det skulle bli att utföra arbetet och skriva om det vi utförde. Vi visste inte heller hur pass komplex databasen var och hur svårt det skulle bli att göra en analys av utdatafilerna. Detta medförde att vi fick ändra lite i våra planer och i slutet kunde vi inte längre följa tidsplanen. Men tack vare de inblandade parterna, vår handledare och uppdragsgivare, lyckades vi nå huvudmålet.

De problem som vi stötte på under arbetets gång var att det inte fanns mycket källmaterial inom projektets område, och det blev svårt för oss att läsa på ämnet.

Under projektets gång har vi fått goda kunskaper i projektarbete och lärt oss att en genomtänkt plan är en viktig del i sådana stora projekt. Vi har även förbättrat vår programmeringsteknik i och med att vi skrev en egen programmodul i PHP. Dessutom har vi fått en erfarenhet i att jobba på ett företag. Om vi skulle göra om projektet skulle vi absolut gjort en mer genomtänkt plan och hela tiden under arbetets gång skriva vad vi har gjort. Det är lätt hänt att man glömmer bort det man har gjort.

Referenser

[1] *Handbok Eyes & Hands Forms version 5.* ReadSoft AB, 2002.

[2] EAN Sverige http://www.ean.se/templates/ean/EanSimplePage____2012.aspx ,2004-11-21.

[3] Prifloat AB. *Prifloat BUSINESS ANALYSIS.* htpp://www.prifloat.com, 2004-11-10.

[4] Scientific Communication. http://www.sciecom.org/links/ETeknik/ TFormat_programvaror_termer_etc/link5382.tkl, 2004-12-21.

[5] phpBB Group. *phpportalen.net.* http://www.phpportalen.net/school.php?id=2 , 2004-12-15.

A Bilaga

```
<?                            // get_content.php är filen som HTML-koden
include("get_content.php"); // jobbar mot, det är där data från utdatafilen
?>                           // läggs in i en PHP variabel ("$filedata").

<pre>
--BEGIN--
<?

foreach ($filearray as $key=>$filedata) if(trim($filedata) != "") {

   $filedata = str_replace(chr(13), "", $filedata); // ersäter noll
tecknet.

   if(is_array($nya)) unset($nya);
   if(is_array($delar)) unset($delar);
   print "----- Svar " . $key . " -----\n"; // visar var en ny enkät börjar.

   $delar = split("#", $filedata);       // överallt där "#" förekopmmer
så.
                                        // delas filedata till en array.
   $nya;                                // $nya = färdiga sträng läggs in.

/***** Variabler som används som räknare **********/

   $items = count($delar);       // räknar antalet platser i $delar.
   $P = 0;                 // $P = indexvariabeln som traverserar i $delar.
   $index = 0;                  // räknare för antalet index frågor.
   $betyg = 0;                  // räknare för antalet betygs frågor.
   $kategori = 0;               // räknare för antalet kategori frågor.

   $alternativ = 0;             // räknare för antalet alternativ frågor.
   $text = 0;                   // räknare för antalet text frågor.

/******* Analyserar och ersätter strängarna som finns i $delar *********/

   while($P < $items)
   {

     /******* känner igen ett namn med hjälp av rubriken "namn" *******/
           if(!(strncmp($delar[$P], "namn", 4)))
           {
              $nya[] = $delar[$P] . ": " . $delar[$P+1] ;
                                        // "namn" => "namn: (svar)".
              $P += 2;          // Ökas till nästa rubrik eller värde.
           }

     /****** känner igen ett email med hjälp av rubriken "email" ******/
           if(!(strncmp($delar[$P], "email", 5)))
           {
              $nya[] = $delar[$P] . ": " . $delar[$P+1];
                                        // "email" => "email: (svar)".

              $P += 2;          // Ökas till nästa rubrik eller värde.
           }

     /******** känner igen en text med hjälp av rubriken "text" ********/
           if(!(strncmp($delar[$P], "text", 4)))
```

```
          {
             $nya[] = $delar[$P] . ++$text . ": " . $delar[$P+1];
                                  // "text" => "text(1,2,..n): (svar)".
             $P += 2;             // Ökas till nästa rubrik ellr värde.
          }

     /******* känner igen en index fråga när värdet 999 dyker upp *******/
          if(!(strncmp($delar[$P], "999", 3)))
          {
             $delar[$P] = substr(chunk_split($delar[$P], 3, ': '), 0, 6);
                                   // sätter ":" mellan identifieraren
                                  // "999" och vikt svaret.
            $delar[$P]= ubstr_replace($delar[$P], "index". ++$index,0, 3);
                                  // "999" => "index(1,2,..n): (vikt)".
             $nya[] = $delar[$P] . "-" . $delar[$P+1];
                                  // index(1,2,..n): (vikt)-(betyg).
             $P += 2;
          }

     /****** känner igen en kategori fråga när värdet 97 dyker upp ******/
          if(!(strncmp($delar[$P], "97", 2)))
          {
             if(!(strncmp($delar[$P], "970", 3)))  // För att ta bort 0 om
             {                                      //svaret är mindre än 9.
                $delar[$P]=substr(chunk_split($delar[$P], 3, ': '), 0, 6);
                                   // sätter ":" mellan identifieraren "97"
                                  // och svaret.
                $delar[$P] = substr_replace($delar[$P], "kategori".
++$kategori,0, 3);
                                  // "97" => "kategori(1,2,..n): (vikt)".
             }
             else
             {
               $delar[$P]= substr(chunk_split($delar[$P], 2, ': '), 0, 6);
                                   // sätter ":" mellan identifieraren "97"
                                  // och svaret.
                $delar[$P] = substr_replace($delar[$P], "kategori".
++$kategori,0, 2);
                                  // "97" => "kategori(1,2,..n): (svar)".
             }
             $nya[] = $delar[$P];
             $P++;
          }

     /******* känner igen en betygs fråga när värdet 998 dyker upp *******/
          if(!(strncmp($delar[$P], "998", 3)))
          {
             $delar[$P] = substr(chunk_split($delar[$P], 3, ': '), 0, 6);
                                   // sätter ":" mellan identifieraren "998"
                                  // och svaret.
            $delar[$P]=substr_replace($delar[$P], "betyg". ++$betyg,0, 3);
                                  // "998" => "betyg(1,2,..n): (svar)".
             $nya[] = $delar[$P];
             $P++;
          }

/*** känner igen en alternativ fråga med hjälp av rubriken "alternativ" **/
          if(!(strncmp($delar[$P], "alternativ", 3)))
          {
             $delar[$P] = substr_replace($delar[$P], "alternativ".
++$alternativ. ": ",0, 3);
```

```
                        // "alternativ" => "alternativ(1,2,..n):(svar)".
            $E = $P;    // för att hålla reda på vart "alternativ" finns.
            $P++;
            if ($delar[$P] < 1000)   // för att försäkra sig om att
            {                         // svaret inte är en rubrik eller
                                     // värde som 999X, 97XX eller 998X.

               $local;
               while($delar[$P] < 1000)   // samlar in alla svar som hör
                                          // till rubriken "alternativ",
                    {                    // och lägger svaren i $local
                                        // variabeln.
                       $local[] = $delar[$P];
                       $P++;
                    }
                    $nya[] = $delar[$E] . implode("-", $local);
                                         // kopplar ihop svaren med rubriken,
            }                           // svaren skills av men "-".
            else{
                    $nya[] = $delar[$E] . $delar[$P];
                           // Om det endast finns ett svar till frågan.
            }
            unset($E);unset($local);
                           // dealokerar dessa lokalt använa variabler.
         }

   }
   $outdata = implode("\n", $nya);  // Hela svaren sätts ihop till en enda
                                   // sträng.
   print $outdata;
}
?>
--END--
```