En objektorienterad, semistrukturerad databas för lagring av proteininformation

(HS-IDA-EA-01-101)

Linus Ahlgren (a98linah@student.his.se)

Institutionen för datavetenskap
Högskolan i Skövde, Box 408
S-54128 Skövde, SWEDEN

Examensarbete på programmet för systemprogrammering under vårterminen 2001.

Handledare: Henrik Jacobsson

En objektorienterad, semistrukturerad databas för lagring av proteininformation

Examensrapport inlämnad av Linus Ahlgren till Högskolan i Skövde, för Kandidatexamen (B.Sc.) vid Institutionen för Datavetenskap.

2001-06-08

Härmed intygas att allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och att inget material är inkluderat som tidigare använts för erhållande av annan examen.

Signerat: _____

En objektorienterad, semistrukturerad databas för lagring av proteininformation

Linus Ahlgren (a98linah@student.his.se)

Sammanfattning

Under det senaste årtiondet har molekylärbiologin genomgått stora förändringar. Genom utvecklingen av tekniker för DNA-sekvensiering har man kunnat utforska stora mängder information. Denna information lagras i biodatabaser som i många fall är länkade med varandra för att ge användaren ett bredare sökfält. SWISS-PROT är ett exempel på en sådan biodatabas som lagrar proteininformation.

Användare som har mindre kunskap om databasens uppbyggnad och unika identifierare kan dock stöta på problem i form av en mängd sökresultat, där endast en mindre del är intressanta för användaren. Missförstånd och tvetydigheter kan också uppstå i länkade biodatabaser, då databaserna har olika uppbyggnad och definitioner på olika ”byggstenar” i databasen.

Arbetet i denna rapport ska därför fokusera på att skapa en databas för molekylärbiologidata som tar hand om dessa problem. För att kunna åstadkomma detta ska ett databashanteringssystem som har funktionaliteter som erbjuder lösningar till dessa problem användas. Den data, i form av proteininformation, som ska lagras hämtas från en SWISS-PROT-domän. I arbetet kommer databashanteringssystemet Lore att användas.

Nyckelord: databas, semistrukturerad data, molekylärbiologi

Innehållsförteckning

1 Inledning

Under det senaste årtiondet har molekylärbiologin genomgått stora förändringar. Detta beror till stor del på utvecklingen av tekniker för DNA-sekvensiering, använda i bland annat genomprojektet HUGO (U.S. Department of Energy, 1997), och inom dator-baserad teknologi (Attwood och Parry-Smith, 1999). Genom att använda dessa tekniker och dator-baserad teknologi finns det möjligheter att hantera och manipulera den information som fås genom bland annat sekvensiering. Information i form av detekterade och analyserade proteinsekvenser ger i vissa fall svar på biologiska frågor som ställs om proteiners struktur och funktionalitet och ibland även evolutionära frågor om olika organismer.

Information om proteiner lagras i olika databaser över Internet. Ett exempel på en sådan biodatabas är SWISS-PROT (Apweiler och Bairoch, 1998), som lagrar proteininformation. Användaren av denna biodatabas kan söka efter en stor mängd information i denna biodatabas genom att specificera ett eller flera sökord. SWISS-PROT, liksom många andra biodatabaser, är länkad till andra biodatabaser, bland annat eftersom de lagrar samma protein, men också för att typen av information om proteinet kan variera i olika databaser. Representationen av den data som ligger lagrad i dessa databaser kan variera. Namn på dataobjekt, objektidentifierare och register kan variera mellan olika databaser. Schulze-Kremer (1998) beskriver exempelvis hur definitionen på högnivåbegreppet *gen* kan variera mellan olika databaser. Dessa skillnader kan orsaka semantiska missförstånd och tvetydigheter för användaren som utnyttjar databaserna och då databaserna utbyter information.

Genom att klart definiera begrepp som är speciellt använda inom molekylärbiologi kan man avhjälpa dessa fel som kan uppstå. Enligt Schulze-Kremer (1998) finns det dock ingen generellt använd begreppssamling, även kallad *ontologi*, och de är inte helt anpassade efter molekylärbiologi. Detta kommer att beskrivas översiktligt senare i rapporten.

Sökning av data i en biodatabas kan underlättas för användaren om den är uppbyggd på begrepp istället för ord. För att underlätta lagringen av data kan man med fördel använda sig av ett databashanteringssystem som kan vara lämpligt för molekylärbiologi. I ovanstående text beskrevs hur molekylärbiologiområdet expanderat på senare tid, till stor del beroende på de tekniker som utvecklats. I och med denna utveckling har man fått större informationsmängder i form av data att lagra. Denna data kan dessutom ändra struktur i och med att nya upptäckter görs, vilket gör att man kanske inte kan lagra denna data på samma sätt som man gjort innan. Dessa egenskaper på data stämmer väl överens med egenskaperna hos semistrukturerad data, vilket gör att ett databashanteringssystem anpassat för semistrukturerad data skulle kunna utgöra ett bra val (se Buneman, 1997, för en introduktion till begreppet semistrukturerad data).

Lore (Abiteboul m.fl., 1997b) är ett databashanteringssystem speciellt anpassat för att hantera semistrukturerad data. Genom att lagra data i en databas i Lore kan användaren genom frågespråket Lorel få tillgång till lagrad data. Detta görs genom ett användargränssnitt i form av en grafisk eller textuell sökmotor. Lore har objektorienterade egenskaper, eftersom den bygger på OEM (Object Exchange Model) (Garcia-Molina m.fl., 1995). I OEM representeras alla entiteter som objekt. Två stora typer av databaser är just den objektorienterade databasen och den relations-orienterade databasen. Den objektorienterade databasen har dock egenskaper som

skiljer den från den relations-orienterade databasen. Ett exempel på detta är att skaparen av databasen kan specificera både strukturen på komplexa objekt och operationerna som kan appliceras på dessa objekt (Elmasri och Navathe, 2000).

I ovanstående text har vi nu beskrivit biodatabasen SWISS-PROT, ontologier och databashanteringssystemet Lore. Användaren av SWISS-PROT kan genom ett eller flera ord söka efter information i databasen. De användare som har mindre information om biodatabasens uppbyggnad eller unika identifierare kan dock få problem då sökningar ska utföras. En stor mängd resultat kan genereras, där endast en mindre del är av intresse för användaren. Biodatabasens länkning kan också innebära problem för användaren. Missförstånd och tvetydigheter kan uppstå då olika biodatabaser kan ha olika uppbyggnad och definitioner på olika ”byggstenar”.

Denna rapport ska med avseende på dessa problem fokusera på att med hjälp av Lore skapa en databas för molekylärbiologidata som helt eller delvis kan eliminera dessa problem. Den data som ska lagras är tagen från en SWISS-PROT-domän och databasen som ska lagra denna data ligger i databashanteringssystemet Lore. Rapporten riktar sig mot läsare med grundläggande kunskaper om relationsdatabaser och objektorientering.

2 Bioinformatik - en kort introduktion

I detta kapitel beskrivs forskningsområdet *bioinformatik* kortfattat. För att läsaren ska kunna förstå och ta in den text som följer definieras här två grundläggande begrepp inom molekylärbiologin; *genom* och *sekvensiering*. De två definitionerna görs på engelska och efter detta följer en fri översättning på svenska.

***Genome*:** All the genetic material in the chromosomes of a particular organism; its size is generally given as its total number of basepairs (Attwood och Parry-Smith, 1999, s. 203).

***Sequencing*:** Determination of the order of nucleotides (base sequences) in a DNA or RNA molecule, or the order of amino acids in a protein (Attwood och Parry-Smith, 1999, s. 208).

En fri översättning till svensk text för definitionen på sekvensiering ger att sekvensiering innebär att fastställa ordningen på nukleotider (bassekvenser) i en DNA eller RNA molekyl, eller ordningen på aminosyrorna i ett protein. Ett *genom* är då, enligt denna definition, allt genetiskt material i en organisms kromosomer där dess storlek oftast är given som dess totala antal baspar.

Enligt Attwood och Parry-Smith (1999) kan bioinformatik idag, fritt översatt till svenska, anses betyda *informationsteknologi för att hantera och analysera biologisk data*. De senaste 20-30 åren har molekylärbiologin dominerats av *strukturell biologi*, exempelvis genom analysering av proteinsekvenser för att kunna fastställa dess uppbyggnad av aminosyror (Attwood och Parry-Smith, 1999). Genom *genomprojekt*, exempelvis HUGO (U.S. Department of Energy, 1997), har man nu ändrat fokus mot sekvensanalys. Genom att rationalisera de stora massorna av sekvensinformation och sedan analysera dessa kan kunskapen inom området bioinformatik öka. Målet med sekvensanalyseringen är att känna igen olika uppbyggnader av sekvenser och byggstenarna i en sekvens samt kanske kunna bygga egna sekvenser genom informationen om byggstenarna. Om man ser sekvensinformationen som ett språk kan detta vara dekomponerat i meningar (proteiner), ord (delar av proteiner) och bokstäver (dess alfabet, aminosyror). Bokstäverna i sig har, enligt denna definition, ingen mening, men i kombination av ord kan de beskriva olika proteiner. Om en bokstav förändras i ett ord kan ett ord förändras och ibland hela meningen. Detta gör att det är viktigt att "dechiffrera" koden korrekt (Attwood och Parry-Smith, 1999).

Med den kunskap man har idag kan man känna igen olika aminosyror och proteiner. Sekvensieringsprocessen är en oerhört komplex process och utvecklandet av denna kan i sig ses som en framgång inom molekylärbiologin. Man har dock inte kunskap för att skapa nya proteiner med hjälp av existerande aminosyror. Detta är ett framtida mål.

2.1 SWISS-PROT

För att kunna utnyttja den information i form av proteinsekvenser som fås genom sekvensiering, kan man använda en *biologisk databas*. Genom att lagra proteinsekvenserna som data i en databas kan man genom olika funktioner manipulera och söka i denna data.

När det gäller strukturen på proteinsekvenser finns det olika nivåer som lagras i olika databaser. Den primära datastrukturen på en proteinsekvens består av 20 olika

aminosyror. Detta innebär att proteinsekvensen skrivs som en linjär följd av bokstäver, där varje bokstav representerar en aminosyra.

SWISS-PROT (Apweiler och Bairoch, 1998) är en databas som lagrar primära proteiner, det vill säga sekvensen av aminosyror i ett protein. Databasen skapades 1986 och underhölls i samarbete mellan Departementet för Medicinsk Biokemi på Geneves Universitet och EMBL- databiblioteket (European Molecular Biology Laboratory). Efter 1994 flyttades samarbetet till EMBL UK Outstation, EBI (European Bioinformatics Institute) och i april 1998 till SIB (Swiss Institute for Bioinformatics). I nuläget underhåller SIB och EBI/EMBL databasen (Attwood och Parry-Smith, 1999).

SWISS-PROT-databasen och dess hemsida strävar efter att ge läsaren av databasen en strukturerad, lättläst och översiktlig bild över de olika proteinsekvenser som är lagrade i databasen. Varje proteinsekvens beskrivs med namn och ett unikt identifikationsnummer. Databasen strävar dessutom efter att förse läsaren av databasen med hög-nivåkommentarer om proteinet, beskrivningar på dess funktioner, strukturen på dess domän, varianter och så vidare (European Bioinformatics Institute, Swiss Institute for Bioinformatics, Söksida över Internet). Databasen är länkad till många andra databaser som också har information om proteiner men med helt annan information, exempelvis 3D-strukturer.

2.1.1 Detaljerad beskrivning av SWISS-PROT

För att läsaren ska få en uppfattning om hur proteiner lagras i SWISS-PROT (Apweiler och Bairoch, 1998) kommer ett exempel på ett protein lagrat i SWISS-PROT att beskrivas i detta kapitel.

Varje protein presenteras i SWISS-PROT med en identitet (**ID**) och ett slut (//). Varje rad markeras med två bokstäver som beskriver vad raden har för innehåll.

Exempel 1. Proteinet 1431_MAIZE beskriven i en SWISS-PROT-domän

```
ID  1431_MAIZE   STANDARD;   PRT;  261 AA.
AC  P49106;
DT  01-FEB-1996 (REL. 33, CREATED)
DT  01-FEB-1996 (REL. 33, LAST SEQUENCE UPDATE)
DT  01-OCT-1996 (REL. 34, LAST ANNOTATION UPDATE)
DE  14-3-3-LIKE PROTEIN GF14-6.
GN  GRF1.
OS  ZEA MAYS (MAIZE).
OC EUKARYOTA; PLANTA; EMBRYOPHYTA; ANGIOSPERMAE;
MONOCOTYLEDONEAE;
OC  CYPERALES; GRAMINEAE.
RN  [1]
RP  SEQUENCE FROM N.A.
RX  MEDLINE; 95148741.
RA  DE VETTEN N.C., FERL R.J.;
RL  PLANT PHYSIOL. 106:1593-1604 (1994).
CC  -!- FUNCTION: IS ASSOCIATED WITH A DNA BINDING COMPLEX TO
BIND TO
```

```
CC    THE G BOX, A WELL-CHARACTERIZED CIS-ACTING DNA REGULATORY
ELEMENT
CC    FOUND IN PLANTS GENES.
CC  -!- SIMILARITY: BELONGS TO THE 14-3-3 FAMILY OF PROTEINS.
DR  EMBL; S77133; G998430; -.
DR  MAIZEDB; 65832; -.
KW  MULTIGENE FAMILY.
SQ  SEQUENCE  261 AA; 29662 MW; C3BB0B1B CRC32;
MASAELSREE NVYMAKLAEQ AERYEEMVEF MEKVAKTVDS EELTVEERNL
LSVAYKNVIG ARRASWRIIS SIEQKEEGRG NEDRVTLIKD YRGKIETELT
KICDGILKLL ETHLVPSSTA PESKVFYLKM KGDYYRYLAE FKTGAERKDA
AENTMVAYKA AQDIALAELA PTHPIRLGLA LNFSVFYYEI LNSPDRACSL
AKQAFDEAIS ELDTLSEESY KDSTLIMQLL RDNLTLWTSD ISEDPAEEIR
EAPKRDSSEG Q
//
```

Exemplet ovan visar en beskrivning av ett protein i en SWISS-PROT-domän. Nedan följer en förklaring på hur de olika raderna ska tolkas med utgångspunkt ur exemplet ovan (Attwood och Parry-Smith, 1999).

ID används för att definiera ett användarvänligt ingångsnamn på proteinet följt av information om storleken på proteinet. I exemplet visat som `1431_MAIZE STANDARD; PRT; 261 AA.` Användarnamnet kan skilja mellan olika databaser, vilket gör att man genom **AC** kan identifiera ett unikt identifieringsnummer som inte ska skilja mellan olika databaser. Detta kan ses som `P49106` i exemplet ovan. **DT** ger information om vilket datum proteinet lades in i databasen och vilka datum det modifierats. Namnet på proteinet ges av ett eller flera namn på proteinet genom **DE**. I exemplet visat som `14-3-3-LIKE PROTEIN GF14-6.` Efter denna information följer information om vilket gennamn proteinet har. Denna rad börjar med bokstäverna **GN** och visas i exemplet som `GRF1`. Bokstäverna **OS** ger sedan information om vilket artnamn proteinet tillhör och **OC** ger organismklassificeringen.

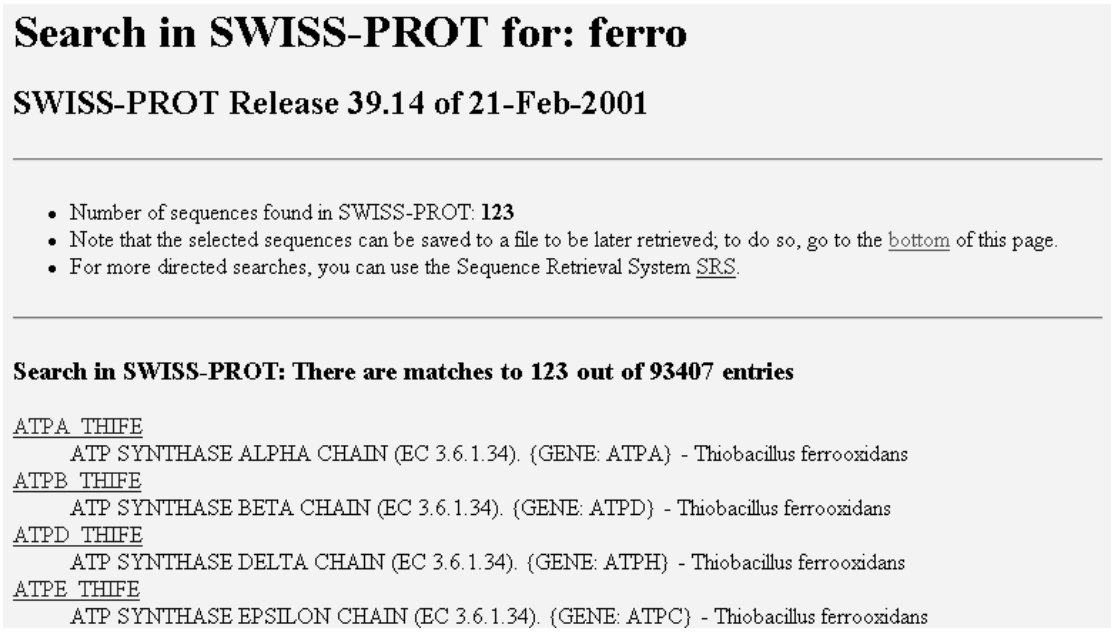
Nästföljande rader börjar på **R** och en bokstav. Dessa rader ger en lista på olika referenser från exempelvis litteratur, opublicerad information från sekvensieringsprojekt, data från strukturella studier eller studier av mutagenitet. **CC** ger ett antal rader med olika kommentarer. Dessa kommentarer delas in i olika teman som beskriver proteinets funktion, dess vävnads-specificering, lokalisering i cellen och så vidare. I de fall information om likheter med andra proteiner är tillgängliga beskrivs detta också.

Om referenser till andra molekylärdatabaser existerar visas detta under **DR**. I detta exempel finns referenser till databasen `EMBL` och `MAIZEDB`. Om relevanta nyckelord till proteinet finns lagrade visas detta under **KW**. I detta exempel visas detta som `MULTIGENE FAMILY.`

Nästföljande rad eller rader, som dock inte finns i detta exempel, börjar med bokstäverna **FT**. Den information som följer efter denna radbeteckning beskriver intressanta regioner i proteinsekvensen, tillsammans med lokala sekundära strukturer, modifieringar som gjorts innan en översättning gjorts och så vidare. Varje rad innehåller ett nyckelord, en beskrivning av var i sekvensen egenskapen ligger lagrad och en kommentar som beskriver hur pålitlig kommentaren är. Sist, men inte minst, kommer själva beskrivningen av proteinsekvensen i form av aminosyror. Detta står efter bokstäverna **SQ**.

2.1.2 Sökning i SWISS-PROT

I kapitlet ovan beskrevs ett exempel på ett protein lagrat i SWISS-PROT (Apweiler och Bairoch, 1998). Genom ett användargränssnitt distribuerat via Internet (European Bioinformatics Institute, Swiss Institute for Bioinformatics, Söksida över Internet) kan användaren söka på ett eller flera ord i databasen. Resultatet kan visas som noll, en eller flera lyckade matchningar. I detta kapitel kommer ett exempel på en sådan sökning att visas, eftersom sökningsmetoden är väldigt komplex.



Search in SWISS-PROT for: ferro

SWISS-PROT Release 39.14 of 21-Feb-2001

- Number of sequences found in SWISS-PROT: **123**
- Note that the selected sequences can be saved to a file to be later retrieved; to do so, go to the bottom of this page.
- For more directed searches, you can use the Sequence Retrieval System SRS.

Search in SWISS-PROT: There are matches to 123 out of 93407 entries

ATPA_THIFE
ATP SYNTHASE ALPHA CHAIN (EC 3.6.1.34). {GENE: ATPA} - Thiobacillus ferrooxidans
ATPB_THIFE
ATP SYNTHASE BETA CHAIN (EC 3.6.1.34). {GENE: ATPD} - Thiobacillus ferrooxidans
ATPD_THIFE
ATP SYNTHASE DELTA CHAIN (EC 3.6.1.34). {GENE: ATPH} - Thiobacillus ferrooxidans
ATPE_THIFE
ATP SYNTHASE EPSILON CHAIN (EC 3.6.1.34). {GENE: ATPC} - Thiobacillus ferrooxidans

Bild 1. En del av resultatet som visas efter sökning på ordet ferro i databasen SWISS-PROT.

I bild 1 ovan visas en del av det resultat som visas då användaren söker på ordet ”ferro” i en SWISS-PROT-domän. Här syns endast fyra av de 123 träffar som matchats mot ordet ”ferro” och i denna sökning är resultatet ordnat i bokstavsordning. En del av övre och nedre del av resultatsidan är inte medtagen på denna bild, eftersom endast en mindre delmängd av resultatsidan räcker för att läsaren ska förstå hur resultatet visas upp. Genom att klicka på en av de understrukna länkarna som beskriver ett specifikt protein, exempelvis ATPA THIFE beskrivet överst på bilden, kan läsaren sedan få detaljerad information om en viss proteinsekvens. Informationen ska i stora drag stämma överens med den information som visas i kapitel 2.1.1.

2.2 Ontologier för molekylärbiologi

I kapitel 2 och 2.1 beskrevs allmänt om *bioinformatik* och databasen SWISS-PROT (Apweiler och Bairoch, 1998) som lagrar proteinsekvenser av primär struktur. Andra databaser och databashanteringssystem lagrar respektive hanterar proteinsekvenser av sekundär och sammansatt struktur. Ibland lagrar flera olika databaser samma typ av information, exempelvis i form av samma proteinsekvenser. De kan då vara länkade med varandra för att en användare av en databas ska få tillgång till information som kanske inte finns lagrad i en databas, men i en annan.

När två databaser är länkade på detta sätt kan tveksamheter och semantiska missförstånd uppstå om databaserna använder egna unika namn eller kategorier eller om de använder identiska namn och kategorier, fast med olika innebörd. Schulze-

Kremer (1998) beskriver begreppet *gen* som ett centralt högnivå-begrepp och hur detta begrepp definieras på skilda sätt i olika databaser.

För att, så långt som möjligt, minska dessa semantiska missförstånd, kan det därför vara bra att samla in en mängd innehållande de viktigaste och mest använda begreppen inom molekylärbiologi. Denna mängd ska sedan vara allmänt åtkomlig för databashanterare, exempelvis via Internet, för att kunna användas för att skapa ett nytt databasschema eller för att beskriva en exakt, schematisk specifikation av begreppen använda i ett existerande schema.

I detta stycke beskrivs ett sätt att skapa sådana definitionsmängder för molekylärbiologi med en *ontologi* (Schulze-Kremer, 1998). Inom datavetenskapen används ontologin som en kortfattad och enkel översikt som beskriver varför vissa entiteter är relevanta i en applikationsdomän och hur de förhåller sig till varandra. Det är inget schema utan en sammanställning av alla nödvändiga "byggblock" med regler för hur och varför entiteter har en relation till varandra. Genom att använda sig av en gemensamt antagen ontologi, kan tveksamheter och missförstånd undvikas och i slutändan kan datorprogram söka efter begrepp i en databas istället för ord.

Cyc (Lenat, 1995) och µKosmos (Mahesh och Nirenburg, 1996) är två ontologier som presenteras i (Schulze-Kremer, 1998) som två ontologier till viss del anpassade för molekylärbiologi. Det finns dock nackdelar med dessa ontologier som gör att de är mindre lämpade för molekylärbiologi. I nedanstående text beskrivs några exempel (Schulze-Kremer, 1998) som visar att dessa ontologier inte är anpassade för molekylärbiologi. För båda dessa ontologier gäller generellt att de inte definierar eller har vaga definitioner för hur begrepp ska klassificeras i subbegrepp. Detta gör att begrepp kan placeras på olika ställen i ontologin beroende på vem som tolkar och hur han tolkar begreppet. I µKosmos är dessutom definitionerna på begrepp avsiktligt vaga för att kunna inbegripa en större mängd naturliga språk. Detta är dock inte passande för molekylärbiologi vars definitioner oftast ska vara exakta. Författarna till Cyc har dessutom angett att de försöker undvika redundans i ontologin, vilket enligt (Schulze-Kremer, 1998) tyvärr dock överskrids ett antal gånger. I ett exempel nedan, som fritt översatt klassernas namn till svenska, beskrivs detta.

Klassen som beskrivs med namnet "Sak", den "universella mängden av allting", har subklasserna som beskrivs med namnen ”Individuellt objekt", ”Ogripbart" och "Roll". Alla dessa tre subklasser överlappar varandra redundant, eftersom det existerar ogripbara, individuella objekt och en roll är någonting som både kan vara individuellt och ogripbart. Detta visar på redundans.

Schulze-Kremer, 1998, föreslår som kontrast till dessa ontologier en ontologi för molekylärbiologi, OMB (Ontology for Molecular Biology). Den är endast en föreslagen ontologi, men kommer ändå att diskuteras eftersom den kommer att användas i den praktiska delen av arbetet. OMB är en intressant ontologi för detta problemområde, eftersom den är inriktad på området molekylärbiologi och inte har de brister som uppmärksammats i Cyc och µKosmos.

OMB samlar, till skillnad från µKosmos och Cyc, ihop begrepp som är nödvändiga för att beskriva biologiska objekt, experimentella procedurer och uträkningar inom molekylärbiologi. Detta innebär dock inte en insamling av all kunskap inom molekylärbiologi och inte heller en förklaring på varje biologiskt fenomen.

Det innebär endast att samla in alla typer av entiteter som molekylärbiologer använder i sitt professionella tänkande och placera passande begrepp på dessa entiteter.

Relationerna mellan entiteterna kan vara av två olika typer. Den första typen beskriver en entitet som kan vara en delmängd av en annan entitet. Den andra typens relation kan beskrivas med sitt namn och orden ”är en” följt av superentiteten. Ett exempel på den andra typens relation, som inte är taget från molekylärbiologin utan endast är med för att förtydliga, är bil som är ett fordon.

Genom att sätta explicita kriterier för hur man subklassificerar varje begrepp, blir definitionen på ett begrepp vägen från dess nod till rotnoden i ontologin.

Ett problem med att skapa en ontologi specialiserad för molekylärbiologi är att identifiera de olika bibetydelser som ligger dolda i det vardagliga språket. Ett exempel på detta är en kromosom av E. coli-bakterien som är en DNA-molekyl som är ett fysiskt objekt (Schulze-Kremer, 1998). DNA-sekvensen, är ett abstrakt objekt som inte ingår i E. coli. Detta gör att DNA och DNA-sekvens kommer att vara olika grenar i den ontologiska grafen. Subklassificering i OMB kräver ibland att utvecklaren går djupare in i olika begrepp.

För att se en sammanfattning av subklassificeringen av begrepp med konkreta exempel på entiteter och relationer hänvisas läsaren till (Schulze-Kremer, 1998).

3 Lore

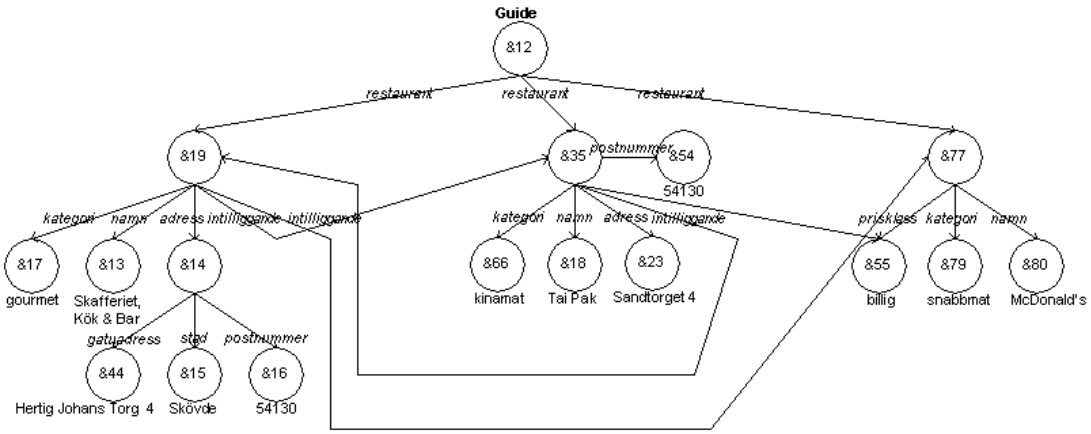
Lore (Abiteboul m.fl., 1997b) är ett databashanteringssystem som utvecklas av en projektgrupp på Stanford-Universitetet. Databashanteringssystemet är speciellt framtaget för att hantera semistrukturerad data. Definitionerna på semistrukturerad data varierar. Buneman (1997) definierar, fritt översatt till svenska, semistrukturerad data som data som kan, men inte behöver, innehålla information om strukturen på lagrad data och där strukturen endast har vaga kopplingar till lagrad data.

Strukturen på data är en av orsakerna till att projektgruppen påbörjade utvecklingen av Lore. En del databasapplikationer kräver att data som ska lagras anpassas till ett schema. Data som är oregelbunden kan vara svår att anpassa till ett sådant schema. Det kan dessutom vara svårt att i förväg bestämma sig för ett schema att följa för data som förändras snabbt och om det är stora mängder data som ska lagras. I föregående kapitel beskrevs SWISS-PROT-databasen (Apweiler och Bairoch, 1998) som också kan sägas vara ett exempel på en databas som hanterar semistrukturerad data. I det fallet biologisk data i form av proteinsekvenser.

I detta kapitel kommer databashanteringssystemet Lore att beskrivas. För att få en övergripande bild av hur Lore är uppbyggt och hur användaren av Lore kan få tillgång till lagrad data beskrivs först den modell som ligger till grund för Lore, OEM (Object Exchange Model) (Garcia-Molina m.fl., 1995), och sedan frågespråket Lorel (Abiteboul m.fl., 1997c). För att ytterligare förtydliga hur man använder sig av Lorel visas ett antal exempel som har sin grund i figur 1 i kapitel 3.1 (Abiteboul m.fl., 1997c).

3.1 Object Exchange Model

Lore är uppbyggt på en datamodell, kallad OEM (Object Exchange Model), som introducerades i Tsimmis-projektet (Garcia-Molina m.fl., 1995). I OEM är alla entiteter representerade som objekt. Man kan se data som representeras i OEM som en graf, med objekt som noder och relationer mellan objekt som kanter mellan noderna. Alla objekt har en unik objektidentifierare (oid). Atomära objekt innehåller ett värde från en av de atomära datatyperna, exempelvis **integer, real, string, gif, html, audio, java** osv. Alla andra objekt är komplexa objekt. Dessa representeras som ett referenspar, betecknat {label, oid}.



Figur 1. En OEM-databas beskriven i form av en graf.

För att användaren ska få åtkomst till ett objekt i en OEM-databas använder man objektnamn. Dessa kan ses som alias för objekt. Det enda sättet för användaren att komma åt objekt via frågor är genom objektnamnen. I exempel 2 nedan visas hur användaren kan få åtkomst till objektet med namnet "Skafferiet, Kök & Bar" visat i figur 1.

Exempel 2. Enkel fråga för att få åtkomst till namnet "Skafferiet, Kök & Bar"

```
select Guide.restaurant.namn
where Guide.restaurant.adress.gatuadress = "Hertig Johans Torg
4"
```

Detta görs genom att med frågespråket Lorel ange objektnamn eller klasser som är åtskilda med punkt. Frågespråket Lorel inklusive fler exempel beskrivs i kapitel 3.2.

3.2 Frågespråket Lorel

Frågespråket som används i databashanteringssystemet Lore heter Lorel (Abiteboul m.fl., 1997c). Genom att ställa frågor i Lorel mot Lore får användare åtkomst till lagrad data.

Lorel är ett användarvänligt frågespråk som liknar frågespråken SQL eller OQL. De stora nyheterna i Lorel, enligt (Abiteboul m.fl., 1997b), som skiljer sig åt från andra frågespråk är

- användandet av kraftfulla sökvägar. Detta är viktigt vid de tillfällen då användaren vet mindre om vilken struktur lagrad data är uppbyggd från.
- en funktion som tvingar två objekt med olika datatyper att jämföras.

3.2.1 Enkla grunder för att använda Lorel och enkla sökuttryck

En av grunderna i Lorel är stödet för enklare sökuttryck då användaren ställer frågor mot lagrad data. Genom att ange de namn som satts på objekt i OEM-modellen i en följd, åtskilda av punkter, kan användaren komma åt objekt som ligger på lägre nivåer. I detta stycke kommer några exempel på frågor att visas i syfte att förklara de enklare grunderna i Lorel och hur enkla sökuttryck kan se ut.

Frågorna som användaren ställer mot lagrad data är i många fall uppbyggda på liknande sätt. Enligt (Abiteboul m.fl., 1997b) använder Lorel en syntax som liknar den som används i frågespråken SQL och OQL. Många frågor som används för att hämta data ur databasen baseras på orden "select" och "from".

Exempel 3. En fråga som returnerar alla restaurantnamn

```
Select Guide.restaurant.namn
From Guide.restaurant
```

I detta exempel hämtas alla namn på restauranter som ligger som subobjekt under objektet restaurant. Frågan resulterar inte i ett fel om det skulle vara så att något objekt restaurant inte har ett subobjekt namn, utan endast de namn som ligger som subobjekt till restaurant returneras som svar till användaren. För att specificera villkor som måste uppfyllas för de objektnamn som returneras använder man ordet "where".

Exempel 4. En fråga som returnerar alla prisklasser > 25 för alla restauranter

```
Select Guide.restaurant.prisklass
From Guide.restaurant
Where Guide.restaurant.prisklass > 25
```

I detta exempel hämtas alla de priser, i detta fall en hänvisning till inom vilken prisklass en restaurant ligger inom, som har ett värde högre än 25. Olika operatorer, i detta fall > ("större än"), kan användas i en where-sats för att specificera frågan ännu mer. Exempel på några av dessa kan vara = ("lika med"), > ("större än"), < ("mindre än"), **and** ("och") och **or** ("eller").

Detta stycke är inte tänkt att beskriva alla de konstruktioner och exempel som kan skapas med utgångspunkt från en specifik databas, utan endast ge läsaren en uppfattning om hur han kan använda enkla sökuttryck för att söka information i en databas. Med orden "**select**", "**from**", "**where**" och operatorer för att specificera frågan kan en stor mängd lagrad data sökas och returneras som svar till användaren. För att underlätta för användaren då han ska specificera en fråga har dock Lorel stöd för generella sökuttryck som beskrivs i nästa stycke.

3.2.2 Generella sökuttryck

Med generella sökuttryck underlättas användarens sökning i databasen. Genom att använda olika uttryck behöver användaren inte känna till den kompletta strukturen på den data som ligger lagrad, vilket gör skrivandet av frågor enklare.

Det första stödet för generella sökuttryck är användandet av *wildcards*. Olika exempel på *wildcards* är **%**, |, **?**, **#**, **like**, **grep** och **soundex**. Det första exemplet, **%**, används för att matchas mot noll eller flera bokstäver i ett namn.

> **Exempel 5.** En fråga som returnerar alla restaurantnamn som har benämningen "billig" och ligger inom postnummerområdet 54130
>
> ```
> Select Guide.restaurant.namn
> Where Guide.restaurant.post% = 54130 and
>    Guide.restaurant.% = "billig"
> ```

I detta exempel vet inte användaren om subobjektet till restaurant är ”post” eller ”postnummer” och använder därför **%** för att matcha alla objekt som börjar på ”post”. Han vet inte heller var objektet prisklass ligger i strukturen, men att det ligger under restaurant. Därför använder han **%** för att matcha alla subobjekt under restaurant mot "billig".

Symbolen **|** används för disjunktion mellan två objektnamn och **?** för att visa att det uttryck som stod skrivet innan **?** är frivilligt. Symbolen **#** matchar ett sökuttryck som inte existerar eller är längre än noll. **Like**-operatorn i Lorel baseras på operatorn med samma namn i SQL.

Det andra stödet för generella sökuttryck är *sökvariabler*. Genom att använda sökvariabler kan användaren ta reda på olika sökvägar och sätta villkor på dessa sökvägar. *Path-of ()* funktionen används för att ta reda på strukturen på lagrad data.

Exempel 6. En fråga som returnerar ett svar i form av alla sökvägar som leder till "postnummer"

```
Select distinct path-of (P)
From Guide.#@P.postnummer


restaurant
restaurant.adress
restaurant.intilliggande
restaurant.intilliggande.adress
```

I exempel 6 ställer användaren en fråga för att ta reda på de olika sökvägar som leder till objektet postnummer. Svaret returneras som alla de vägar som leder till postnummer-objekt.

Fler exempel på hur man kan använda generella sökuttryck för att skapa olika frågor finns i (Abiteboul m.fl., 1997c).

3.2.3 Uppdatering av frågeplaner och lagring av större volymer

I föregående kapitel har vi diskuterat hur användaren av Lorel kan hämta data ur databasen med hjälp av select-satser. I detta kapitel ska vi diskutera översiktligt Lorel's uppdateringsspråk, med vilket man kan skapa eller ta bort databasnamn, atomära och komplexa objekt, modifiera värden och lagra större volymer i en OEM-databas. Uppdateringsfunktionen kan delas upp i tilldelningar av namn till objekt, skapande av objekt och uppdateringar av objekt. Namn är ingångar till databasen. De skapas genom att använda uttrycket **name**.

Exempel 7. Skapande av en ingång till restauranten "Tai Pak"

```
name myFavorite := element
       (Select Guide.restaurant
       Where Guide.restaurant.namn = "Tai Pak")
```

I exemplet ovan skapas en ingång till restauranten "Saigon" som sedan kan användas för att modifiera data i databasen. Namnet `myFavorite` kan sedan tilldelas ett null-värde för att tas bort från databasen. För att skapa nya objekt i databasen använder man sig av funktionen *new_oem.*

Exempel 8. Skapande av ett nytt komplext OEM-objekt

```
new_oem (complex,struct (a:{new_oem (int, 5)}, b:{X,Y}))
```

I exemplet ovan skapas ett nytt komplext objekt med ett godtyckligt namn. Namnet på objektet kan definieras i efterhand. I exemplet skapas sedan en förbindelse mellan detta objekt och ett nytt heltalsobjekt benämnt a med värdet 5. Detta visas i exemplet som *new_oem (int, 5).* Sist sätts två förbindelser till objekten X och Y med namnet b. Funktionen *struct ()* används för att specificera var i ett redan existerande schema objektet ska placeras. För att uppdatera värden på redan existerande objekt använder man ordet "update".

Exempel 9. Uppdatering av objektet prisklass till värdet 7

```
update prisklass := 7
```

I exemplet ovan uppdateras och ändras värdet på heltalsobjektet prisklass till 7. Om man väljer att använda sig av uttrycket **name** istället för **update** skapas först ett nytt

objekt prisklass som sedan övertar det gamla objektet ”prisklass”s roll. För att skapa flera objekt samtidigt använder man sig ordet **load**. Uttrycket är av formen "**load** <filename>", där filename är namnet på filen där alla de uttryck i form av skapande av objekt är skrivna.

Det finns fler exempel på hur man använder sig av **update** och **load** för att uppdatera atomära och komplexa objekt, samt skapa och uppdatera objekt. Dessa ska vi dock inte titta mer på i detta kapitel (se Abiteboul m.fl., 1997c, för mer information). Det kan kommenteras att objekt skapade i Lorel explicit inte kan tas bort. Däremot blir objekt borttagna implicit då de ej kan nås genom någon sökväg.

3.2.4 Tvingande jämförelser mellan olika objekt och/eller värden

En av de främsta egenskaperna i frågespråket Lorel, förutom de kraftfulla sökuttrycken, är funktionen med tvingande jämförelser mellan olika objekt och/eller värden (Abiteboul m.fl., 1997c). Om användaren ställer en fråga mot Lore som innehåller jämförelser mellan olika datatyper ska detta inte resultera i ett fel. Eftersom Lorel har ett så brett användningsområde med både enkla och generella sökuttryck är tvingande jämförelser också nödvändigt för att minska omfattningen på de fel som kan uppstå då en "felställd" fråga ställs mot lagrad data. Två områden med tvingande jämförelser är mellan värden och atomära objekt och mellan objekt och objektmängder. Inom området där man tvingar jämförelser mellan objekt och objektmängder finns dessutom underkategorier. Dessa underkategorier är tvingande jämförelser mellan

- objekt och objekt
- värden, atomära objekt, objektmängder och komplexa objekt
- objektmängder och objektmängder
- värden, atomära objekt och objektmängder

Hur jämförelsen rent tekniskt går till kommer inte att diskuteras i detta kapitel. För läsaren är det däremot viktigt att veta att Lorel utför tvingande jämförelser både för att underlätta för användaren av Lorel och för att undvika fel som kan störa användaren då han söker i databasen.

3.3 Systemarkitektur

Åtkomst till Lore systemet kan man få på två olika sätt. Det första är att använda sig av en mängd olika applikationer och det andra genom att använda ett av två olika gränssnitt. Det ena är ett textgränssnitt som främst används av systemutvecklaren och det andra är ett grafiskt gränssnitt som användaren utnyttjar. I detta avsnitt beskrivs Lore’s systemarkitektur. Beskrivningen är tagen från (Abiteboul m.fl., 1997b).

Det grafiska gränssnittet förser användaren med olika funktionaliteter såsom

- ett kraftfullt verktyg för att bläddra, skumma igenom frågeresultaten
- en dataguide för att få en överblick av databasstrukturen och kunna formulera enkla frågor
- ett sätt att lagra frågor som ställs ofta
- en mekanism för att kunna se multimedia uppbyggt på datatyper såsom video, audio och java

De två gränssnitten kommunicerar med Lore genom ett API (Application Programming Interface).

Detta API kommunicerar i sin tur med ett s.k. *frågekompileringslager* med olika funktionaliteter. *Frågekompileringslagret* i Lore består av en *mekanism för analysering av data*, en *förprocessor*, en *frågeplansgenerator* och en *frågeoptimerare*.

Mekanismen för analysering av data tar emot en textuell representation av en fråga ställd av användaren från API. Den gör sedan om denna representation i form av ett analysträd och skickar sedan detta vidare till *förprocessorn*.

Förprocessorn transformerar frågan ställd i Lorel till en liknande fråga i OQL (Object Query Language) och *frågeplansgeneratorn* genererar sedan en frågeplan som den skickar vidare till *frågeoptimeraren*.

Frågeoptimerarens funktion är att göra vissa optimeringar i frågeplanen samt bestämma om användningen av index är lämpligt. Den optimerade frågeplanen skickas sedan till datalagret.

Datalagret består av en *OEM-objekthanterare, frågeoperatorer, en extern datahanterare* och vissa andra användbara enheter, såsom *dataguider*. *Frågeoperatorerna* exekverar den genererade frågeplanen. Exempel på frågeoperatorer är **scan**, **select**, **join** och **project**. Dessa operatorer och ytterligare operatorer kommer inte att tas upp i denna rapport, men för den intresserade läsaren hänvisas till artikeln (Abiteboul m.fl., 1997b). I denna artikel står också en introduktion till hur de används. *OEM-objekthanteraren* fungerar som ett översättningslager mellan OEM (Object Exchange Model) (Garcia-Molina m.fl., 1995) och lägre nivåer. Den har funktionaliteter som att hämta objekt, jämföra två objekt, utföra tvingande jämförelse mellan två objekt av olika datatyper och så vidare. Tvingande jämförelser beskrevs översiktligt i kapitel 3.2.4.

Den externa datahanteraren hämtar information från externa datakällor vid frågeprocessen. En *dataguide* används för att ge en summering av strukturen på en OEM-databas.

Den externa datahanteraren förser Lore med data från externa källor. Detta är användbart exempelvis i situationer där data som efterfrågas av användaren inte ligger lagrat i Lore´s databas, men lokaliseras av datahanteraren från en extern källa. Denna data lagras då som ett externt objekt i Lore´s databas under den tid användaren efterfrågar data och kopplingen samt objektet tas sedan bort då frågeexekveringen är slutförd. Objektet är inte synligt för användaren utan ligger tillsammans med data hämtad från databasen.

En liten databas med insamlad information från externa källor kan på detta sätt ses ur två perspektiv, den fysiska och den logiska vyn. Den logiska vyn ses av användaren som en helhet där all data, både intern och extern, är samlad. Den fysiska vyn visar hur Lore, genom att använda **scan**-operatorn, kodar både extern och intern data. Implementationen av den fysiska vyn inkluderar lokaliseringen av ett *Wrapper*-program som hämtar extern data och översätter den till OQL (Object Query Language), ett *kvantum* som indikerar det tidsintervall insamlad data används och en mängd argument som används för att begränsa mängden information som hämtas vid ett anrop.

Om många anrop till en extern datakälla skulle inträffa samtidigt används olika tekniker för att särskilja dessa. Detta är viktigt för att anropen inte ska ta för mycket

av frågeprocessens sammanlagda exekveringstid. Två tekniker för att reducera tiden för frågeprocessen är att sålla bort fler anrop än ett och om tidigare frågors svar kan användas av nuvarande fråga återanvända denna data.

En dataguide används för att ge en summering av strukturen på en OEM-databas. Detta är viktigt för användaren, eftersom en Lore-databas inte har något explicit schema att utgå ifrån och därför måste använda sig av frågeformulering och frågeoptimering i så stor utsträckning som möjligt. Utan en aning om strukturen på en databas blir det svårt för användaren att ställa meningsfulla och lämpliga frågor. Dataguiden fås dynamiskt och innehåller de objektnamn och relationer mellan objekt som ingår i databasen. De är också viktiga för systemet för att kunna lagra statistik och för guidning vid frågeoptimering.

För att få en utförligare beskrivning av den externa datahanteraren och dataguider, se (McHugh och Widom, 1997) respektive (Goldman och Widom, 1997).

3.4 Användningen av Lore för att hantera XML-data

Användningen av Internet har ökat de senaste åren i och med att fler människor får tillgång till datorer via företag eller persondatorer i hemmet. I och med att användningen av Internet ökar ställs det högre krav på de applikationer som används. Utvecklarna av applikationerna vill ha en struktur som är anpassad efter just deras applikation och användarna vill både kunna ta emot information, men också delge information via applikationen till exempelvis en databas.

För att uppfylla dessa krav har märkordspråket XML tagits fram. Det är en ny standard för datarepresentation och utbyte över Internet som bygger på märkordspråket SGML och till viss del på märkordspråket HTML.

De senaste åren har projektgruppen för Lore utvecklat Lore för att hantera XML-data. Detta har man gjort, eftersom man har sett likheter mellan semistrukturerade modeller och XML. Hur överföringen från att hantera semistrukturerad data till att hantera XML har gått till kommer inte att tas upp i detta kapitel, eftersom utveckling av Lore till att hantera XML-data inte har med detta projekt att göra. Det är dock viktigt att ta upp att Lore nu utvecklats till att hantera XML-data och inte semistrukturerad data.

I detta arbete kommer därför en tidigare version av Lore att användas vid framtagandet av en databas, då senare framtagna versioner är anpassade för XML.

4 Problembeskrivning

I det här kapitlet presenteras en problemdomän med avseende på kombinationen biodata och databaser samt en problemavgränsning som detta arbete kommer att fokusera på.

4.1 Biodata och databaser

I kapitel 2 gjordes en allmän beskrivning av bioinformatikområdet och biodatabasen SWISS-PROT (Apweiler och Bairoch, 1998). Efter detta följde ett kapitel med en beskrivning av databashanteringssystemet Lore (Abiteboul m.fl., 1997b).

Valet av SWISS-PROT som diskussionsämne i rapporten bygger på olika problem som kan uppstå då användaren söker efter information i biodatabasen. Problemen och förslag på lösningar till dessa ges i punktform nedan:

- I vissa fall saknar användaren av SWISS-PROT information om unika egenskaper som definierar information om ett specifikt protein (information lagrat under beteckningen "AC"). I dessa fall kan användaren söka i databasen genom enstaka ord, vilket dock genererar en mängd resultatträffar där bara en mindre del är av intresse för användaren. I Lore kan användaren genom frågespråket Lorel använda sig av generella sökuttryck för att finna en mindre mängd resultat, som motsvarar den ställda frågan.
- SWISS-PROT lagrar data efter en viss struktur. Denna struktur beskrivs på SWISS-PROTs hemsida, men kan vara ganska svår att hitta. Detta gör att specifika sökningar i databasen kan bli svåra att utföra, då användaren kanske inte vet vad som unikt skiljer olika proteiner. Detta problem kan lösas i Lore genom en dataguide som beskriver databasens struktur. Användare som inte känner till detta kan ha svårt att ställa meningsfulla och lämpliga frågor i frågespråket Lorel.
- SWISS-PROT är länkad till andra biodatabaser. Dessa databaser kan vara uppbyggda på olika sätt beroende på hur utvecklaren av databasen har strukturerat lagringen av biodata och vilka namn och definitioner som sätts på "byggstenarna". Typen av databas och databashanteringssystem kan också variera. Detta kan orsaka missförstånd och tvetydigheter då användaren söker i SWISS-PROT. Genom att använda en generellt använd ontologi vid skapandet av en databas kan dessa problem reduceras. Om en databas för molekylärbiologidata ska skapas kan det därför vara intressant att exempelvis titta på ontologin OMB (Ontology for Molecular Biology) som är anpassad för just detta område.

4.2 Problemavgränsning

I föregående kapitel beskrevs problem inom SWISS-PROT samt några förslag på lösningar till dessa problem. Det föreslogs bland annat att man skulle kunna använda Lore för lagringen av molekylärbiologidata för att lösa problemen samt en ontologi, OMB, för skapandet av en databas för molekylärbiologidata. Denna data skulle rimligtvis vara proteininformation från SWISS-PROT, eftersom denna biodatabas ligger som grund till problemställningen.

Baserat på dessa problem och förslagen på lösningar till dessa problem kommer därför arbetet i denna rapport att fokusera på att skapa en objekt-orienterad,

semistrukturerad databas för lagring av proteininformation. Data kommer att hämtas från en SWISS-PROT-domän och lagras som en databas i databashanteringssystemet Lore.

5 Metoder och metodval

I detta kapitel kommer ett antal metoder att beskrivas som skulle kunna vara lämpliga för det valda problemområdet. På detta följer sedan en metodspecificering för de metoder som valts för arbetet i rapporten.
Den typ av undersökning som kommer att genomföras i detta projekt är en hypotesprövande undersökning. Patel och Davidsson (1994) anser att denna typ av undersökning vanligen utförs när kunskapsmängden är omfattande och teorier har börjat utvecklas. Om kunskapen är tillräcklig är det möjligt att utifrån dessa teorier skapa antaganden om förhållanden i verkligheten. Sådana antaganden är hypoteser. Undersökningen måste då läggas upp på ett sätt som medför att risken för att något utanför hypotesen påverkar resultatet minskar. För insamlandet av information används en metod som ger en så exakt information som möjligt.

Problemet i denna rapport bygger på en arbetshypotes som säger att det går att skapa en databas i för lagring av proteininformation i databashanteringshanteringssystemet Lore (Abiteboul m.fl., 1997b). Arbetshypotesen är den hypotes som rapporten grundas på.

5.1 Metoder för problemområdet

För att kunna genomföra arbetet i rapporten används olika metoder. Dessa metoder kan ses som arbetssteg i genomförandet. I detta kapitel beskrivs några av dessa metoder.

5.1.1 Kravinsamling

Som första steg i arbetet görs en kravinsamling av de krav som ska ställas på den färdiga databasen. I detta arbete används kraven tillsammans med en testning för att se om arbetshypotesen kan verifieras eller falsifieras. Om testningen visar att kraven inte kan uppfyllas, så kan inte heller arbetshypotesen verifieras. Testningen beskrivs i kapitel 5.1.4.

För insamlingen av krav kan exempelvis en mindre litteraturstudie genomföras. Litteraturstudien kan, genom studerande av böcker och artiklar av framstående forskare inom databasområdet, generera krav som kanske är generellt använda och som i tidigare arbeten visat på ett bra resultat. Böcker som beskriver utvecklingen och skapandet av databaser skulle kunna vara användbara i detta arbete. Problemet med denna litteraturstudie skulle kunna vara att välja ut ett mindre antal böcker eller artiklar för användande i rapporten. Databasområdet är stort och väldigt utvecklat, vilket gör att det finns en stor mängd litteratur att välja bland. Ett annat problem kan vara att finna litteratur som beskriver krav som ställs vid utvecklande av en databas för lagring av molekylärbiologidata, vilket stämmer väl överens med problemet i denna rapport.

Kravanalysen kan också tas fram på egen hand med grund i problemdomänen. Denna metod är användbar i detta arbete, eftersom det kan vara svårt att finna litteratur som beskriver framtagandet av krav vid skapandet av en databas för molekylärbiologidata. Analysen i detta arbete måste ställa krav på databasen som beskriver att information lagras samt att ingen information förloras vid lagringen av informationen. Det är också intressant att titta på om användaren av databasen kan söka efter information i databasen samt hur detta förfarande går till.

5.1.2 Skapande av modell

Efter genomförandet av kravanalysen skapas en modell över hur den tänkta databasen ska se ut. Användaren av databasen får på detta sätt en överblick över databasen och en inblick i hur han får åtkomst till lagrad data.

Det finns olika sätt att visa upp modellen. Ett sätt skulle kunna vara att visa upp modellen som en 3-dimensionell figur tillsammans med en tabell över entiteterna i databasen. Denna metod skulle dock vara att undervärdera databasanvändarens tankeförmåga, då modellen redan innan är väldigt enkel i sitt utförande. I detta arbete kommer modellen av databasen att visas upp som en enkel bild som representerar information som objekt.

5.1.3 Överföring av data

Den data som ska lagras i en databas i Lore (Abiteboul m.fl., 1997b) hämtas från en SWISS-PROT-domän. Lore accepterar endast lagring av data som en fil i OEM-format (Garcina-Molina m.fl., 1995). Detta innebär att två moment måste genomföras:

- Överföring av data från domänen till en fil
- Lagring av filen i Lore

Överföringen av data till en fil kan ske på några olika sätt. Ett av dessa är att manuellt skriva en fil som jämförs med informationen lagrad i domänen. På detta sätt kan man kontrollera antalet överförda proteiner på ett enkelt sätt och helt förlita sig sin egen överföring. Det är dock väldigt enkelt att förvilla sig bland raderna i domänen och på så sätt glömma att överföra viss information eller föra över information till fel ställe i filen.

Ett annat sätt att föra över information är att göra en implementation av ett program i ett specifikt programmeringsspråk. Detta program tar då hand om denna del istället för att göra det manuellt. Enligt Patel och Davidsson (1994) bör det finnas ett tydligt syfte för att motivera att en implementation skall genomföras. Syften kan i detta fall exempelvis vara att visa att någonting går att utföra eller att en implementation är nödvändig för det fortsatta experimentet. Implementationen är nödvändig i detta arbete, eftersom man utan en fil inte kan skapa en databas vilket omöjliggör hela arbetet i rapporten. Programmet kommer att skapas i programmeringsspråket C++ (Lippman, 1993).

5.1.4 Testning av databasen

Som sista del i arbetet kommer en testning av databasen att genomföras. Denna testning kontrollerar de krav som ställts på databasen. Om kraven uppfylls så kan arbetshypotesen verifieras.

I detta arbete skulle en testning kunna utföras för att kontrollera databasens utformning och uppbyggnad. Detta skulle kanske kunna ge svar på om information tappats vid överföringen mellan domänen och databasen. En annan testningsmetod skulle kunna vara att endast titta på användningsområdet för databasen. Metoderna kompletterar varandra väldigt bra, men klarar inte att stå för sig själva vilket gör att en kombination av metoderna kommer att användas i detta arbete. Testningen av databasen kommer att genomföras som en analys av resultatet för att se om arbetshypotesen ska verifieras eller falsifieras.

5.2 Metodspecificering

I ovanstående kapitel diskuterades de metoder som skulle kunna användas i arbetet beskrivet i denna rapport tillsammans med motiveringar för valda metoder. De valda metoderna beskrivs i punktform nedan:

- En kravanalys genomförs för att se vilka krav som ska ställas på databasen. Dessa krav tas fram med utgångspunkt i problemdomänen.
- Genom att skapa en modell över databasen får användaren av databasen en enkel och översiktlig bild. I rapporten visas detta som en enkel bild med information representerad som objekt.
- Överföring av proteininformation från SWISS-PROT-domänen till en databas i Lore genomförs med ett program skapat i programmeringsspråket C++.
- En testning som testar både databasens utformning och användningsområdet för databasen resulterar i en verifiering eller en falsifiering av arbetshypotesen.

6 Genomförande

I detta kapitel kommer det arbete som utförts vid skapandet av en biodatabas att beskrivas. I samband med detta beskrivs även detaljer som kanske inte kunde utföras på tänkt sätt och vilka fel som gjordes. Båda dessa aspekter följs av en förklaring på hur problemet löstes.

Arbetet som beskrivs i kapitlet kan delas upp i olika delkapitel som beskriver de faser som arbetet var uppbyggt på. Arbetet började med en insamling av databaskrav som beskrivs i kapitel 6.1. Detta följdes av skapandet av en modell över hur databasen skulle se ut, beskrivet i kapitel 6.2. I detta kapitel beskrivs också vilka egenskaper ur ontologin OMB (Ontology for Molecular Biology) (Schulze-Kremer, 1998) som använts för att lösa problemet samt vilka som inte använts. I samband med detta beskrivs också varför dessa egenskaper använts samt varför de som inte tagits med inte använts. Efter detta följde skapandet av kod i programmeringsspråket C++. Huvuduppbyggnaden av programmet kommer att beskrivas i kapitel 6.3. I detta kapitel beskrivs också lagringen av data i databasen i Lore (Abiteboul m.fl., 1997b). Slutligen beskrivs i kapitel 6.4 det resultat som framkom vid skapandet av databasen i Lore.

6.1 Kravanalys på databasen

Som första steg i skapandet av databasen görs en mindre kravanalys. I större projektarbeten kan kravanalysen vara första steget i en analysfas. I sådana arbeten brukar denna fas följas av en designfas, en implementationsfas och en testfas. Arbetet som skall utföras i denna rapport är dock ett implementationsarbete och mindre vikt kommer därvid att läggas på analys- och designfaserna, medan däremot fokuseringen kommer att ligga på implementation- och testningsfasen.

De krav som ställts på den framtagna biodatabasen är anpassade till detta problemområde och är tänkta att kunna testas inom den tidsram som arbetet ska pågå. Kraven är de krav som bör vara de främsta då denna databas skapas, det vill säga krav som påverkar databasens funktionalitet och användbarheten av databasen. De framtagna kraven räknas upp i punktform nedan:

- Biodatabasen ska bestå av hela eller en del av de proteiner som lagrats i den ursprungliga SWISS-PROT-domänen
- Den information som ligger lagrad i SWISS-PROT-domänen ska, för de proteiner som lagrats i biodatabasen, vara fullt representerade i biodatabasen
- Det ska gå att genomföra en sökning efter information för ett specifikt protein i biodatabasen och ett korrekt svar ska genereras.
- Det ska gå att genomföra en sökning efter information som omfattar en mängd proteiner i biodatabasen och ett korrekt svar ska genereras.

Det är författarens avsikt att så långt som möjligt uppfylla dessa krav på databasen. Om dessa krav inte uppfylls ska en förklaring till detta ges och en beskrivning på hur problemet lösts eller varför problemet inte lösts i de fall det inte går att lösa.

6.2 Skapande av en modell över databasen.

Efter att kravinsamlingen utförts kan man börja titta på vilken modell data lagrat i Lore (Abiteboul m.fl., 1997b) ska vara uppbyggt kring. Med den kunskap som finns om databashanteringssystemet Lore kan en enkel objektorienterad modell ställas upp. Denna modell skall representera data lagrat i Lore på ett enkelt och användarvänligt sätt samtidigt som den inte förlorar någon av den information som lagrats i SWISS-PROT-domänen.

6.2.1 Användandet av OMB för att ta fram en lämplig modell

I kapitel 2.2 beskrivs översiktligt en ontologi som kallas OMB (Ontology for Molecular Biology) (Schulze-Kremer, 1998). Skaparen av denna ontologi har anpassat denna för just molekylärbiologidata. Denna ontologi nämndes sedan i kapitel 5 som en tänkbar del i lösningen till problemet beskrivet i denna rapport.

Genom analysering av OMB med dess struktur och relationer har författaren kommit fram till att det som kan tänkas vara användbart för problemet i denna rapport är de två relationstyper som beskrivs i OMB. För att förtydliga dessa beskrivs två exempel med entiteter som är godtyckliga och generella. I det första exemplet har man entiteterna fordon och bil, där bil är en subentitet till fordon. Med denna modellering ser man snabbt att en bil är ett fordon. De ord som representerar denna relation är "är en" eller "är ett". I det andra exemplet har man entiteterna bil och däck, där däck är en subentitet till bil. Denna modell ger att bilen består av ett däck. De ord som representerar denna relation är "består av".

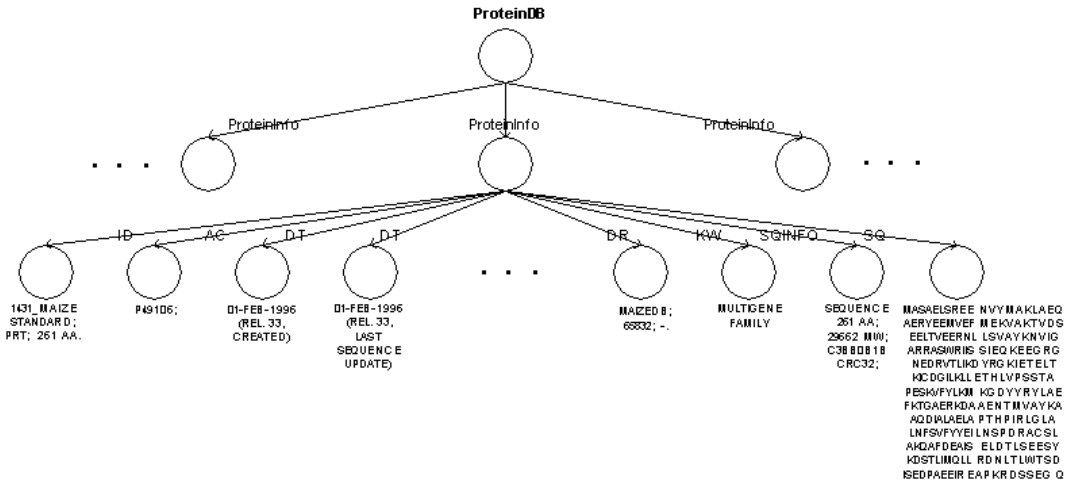
När den första modellen av biodatabasen skapades var det tänkt att det objekt som representerade ett protein skulle benämnas ”Protein”. Under detta objekt skulle sedan information rörande proteinet läggas som ett antal subobjekt. Relationen mellan ”Protein” och subobjekten skulle vara en av de två relationstyperna beskrivna i OMB. Efter att en genomgång gjorts där man definierade de subobjekt som skulle ligga under ”Protein” framkom det dock att flera av dessa inte kunde beskrivas med en av dessa två relationstyper. Ett exempel på detta är kommentarer om ett protein som ligger som information under subobjektet ”CC”. Relationen mellan protein och kommentaren om ett protein kan inte beskrivas med någon av de två relationstyperna beskrivna i OMB.

Efter övertänkande framkom det dock att om man döpte om objektet ”Protein” till ”ProteinInfo”, så kunde alla relationer mellan ”ProteinInfo” och dess subobjekt beskrivas i form av relationstypen som identifieras med orden ”består av”.

Med hänseende till detta kommer den tänkta modellen av biodatabasen att använda sig av denna relation. I nedanstående text beskrivs hur modellen i detalj kommer att se ut.

6.2.2 Modell över Lore-databasen

I detta kapitel beskrivs slutligen den struktur modellen har fått beroende på vilken information som ska lagras och vilka resurser man har att utgå från. I förra kapitlet beskrevs de egenskaper i ontologin OMB, de två relationstyperna, som skall användas för att skapa modellen. Eftersom Lore är en objektorienterad modell måste modellen också vara objektorienterad. Figuren på nästa sida visar en del av denna modell med en del av den information som finns om ett protein.



Figur 2. En del av modellen över Lore-databasen. Denna figur visar en del av den information som beskrivs i exempel 1 i kapitel 2.1.1.

På figuren visas ingångsnamnet till databasen som "ProteinDB". Genom detta namn får användaren åtkomst till all information som ligger lagrad i databasen. Under varje objekt "ProteinInfo" ligger sedan all information för ett unikt protein grupperat. Den information som ligger lagrad under de subobjekt som ligger grupperade under ”ProteinInfo” är endast en del av informationen för ett protein.

Denna modell visar en del av den information som finns lagrad i proteinet beskrivet i exempel 1 i kapitel 2.1.1. Ibland saknar ett protein viss information, exempelvis gennamnet för densamma. Denna information ligger under subobjektet ”GN” och i det fall informationen saknas finns inte detta objekt representerat för detta protein i databasen. För att få en full beteckning och beskrivning över de olika subobjekt som kan ligga under ”ProteinInfo” hänvisas läsaren till kapitel 2.1.1.

Varje proteinsekvens har en unik identifierare som skiljer den från andra proteiner. Den information som representerar denna identifierare ligger lagrad under objektet ”AC”. Genom att skriva ut vägen från objektet ”ProteinDB” till objektet ”AC” kan användaren få åtkomst till information om en specifik proteinsekvens. För att förtydliga detta visas ett exempel, med hjälp av modellen ovan, där användaren vill ta reda på den fullständiga proteinsekvensen, i form av en rad av aminosyror, för en proteinsekvens. Den unika identifieraren för proteinet ligger lagrad under objektet ”AC” och är P49106. Denna information måste användaren veta för att kunna ställa frågan mot databasen. Den fullständiga proteinsekvensen ligger lagrad under objektet ”SQ” och är den information användaren vill få ut. Detta exempel visas nedan.

Exempel 10. Åtkomst till sekvensen för proteinet med AC = P49106

```
Select ProteinDB.ProteinInfo.SQ
Where ProteinDB.ProteinInfo.AC = "P49106";
```

6.3 Skapande av fil och lagring av data i databasen

Detta kapitel beskriver de steg som genomförts för att överföra data från SWISS-PROT-domänen till att lagra denna data i databasen i Lore (Abiteboul m.fl. 1997b). Kapitel 6.3.1 beskriver överföringen av data från SWISS-PROT-domänen till en fil i OEM-format. Se kapitel 6.3.1 för en förklaring till detta. I kapitel 6.3.2 beskrivs sedan förfarandet vid lagring av filen i databasen i Lore.

6.3.1 Skapande av fil i OEM-format

Den data som ligger lagrat som proteinsekvenser i SWISS-PROT-domänen måste omstruktureras för att kunna lagras i Lore. Databashanteringssystemet godtar endast data som lagrats i ett så kallat OEM-format i en fil. I kapitel 2.1.1 visas i exempel 1 hur en proteinsekvens representeras i SWISS-PROT-domänen. En del av detta exempel visas nedan följt av ett exempel på hur det ser ut i OEM-format. Lägg märke till att ”SQ” i exempel 11 indelats i subobjekten ”SQINFO” och ”SQ” i exempel 12 för att särskilja information om sekvensen och sekvensen beskriven i aminosyror. Modellen i föregående kapitel visar ingångsnamnet till databasen som objektet med namnet ”ProteinDB”. Detta objekt har sedan subobjekten ”ProteinInfo”. Under det senare ligger sedan subobjekt som är specifika för ett protein. Subobjekten ”SQINFO” och ”SQ” är två av dessa subobjekt.

Exempel 11. En del av av ett proteins proteininformation representerad i SWISS-PROT-domänen. Hela exemplet visas i exempel 1 i kapitel 2.1.1

```
ID  1431_MAIZE   STANDARD;   PRT;  261 AA.
AC  P49106;
.
.
.
SQ  SEQUENCE  261 AA; 29662 MW; C3BB0B1B CRC32;
MASAELSREE NVYMAKLAEQ AERYEEMVEF MEKVAKTVDS EELTVEERNL
LSVAYKNVIG ARRASWRIIS SIEQKEEGRG NEDRVTLIKD YRGKIETELT
KICDGILKLL ETHLVPSSTA PESKVFYLKM KGDYYRYLAE FKTGAERKDA
AENTMVAYKA AQDIALAELA PTHPIRLGLA LNFSVFYYEI LNSPDRACSL
AKQAFDEAIS ELDTLSEESY KDSTLIMQLL RDNLTLWTSD ISEDPAEEIR
EAPKRDSSEG Q
//
```

Exempel 12. Proteinsekvensen från exempel 10 beskriven i OEM-format

```
<ProteinDB :: ProteinDB {
      <ProteinInfo {
            .
            .
            .
            } >
      <ProteinInfo {
                  <ID "1431_MAIZE   STANDARD    PRT  261 AA.">
                  <AC "P49106">
                  .
                  .
                  .
```

```
                        <SQINFO "SEQUENCE  261 AA  29662 MW  C3BB0B1B
                        CRC32 ">
                         <SQ "MASAELSREE NVYMAKLAEQ AERYEEMVEF
                        MEKVAKTVDS EELTVEERNL LSVAYKNVIG ARRASWRIIS
                        SIEQKEEGRG NEDRVTLIKD YRGKIETELT KICDGILKLL
                        ETHLVPSSTA PESKVFYLKM KGDYYRYLAE FKTGAERKDA
                        AENTMVAYKA AQDIALAELA PTHPIRLGLA LNFSVFYYEI
                        LNSPDRACSL AKQAFDEAIS ELDTLSEESY KDSTLIMQLL
                        RDNLTLWTSD ISEDPAEEIR EAPKRDSSEG Q">
              } >
              <ProteinInfo {
              .
              .
              .
              } >
} >
```

För att kunna överföra data från SWISS-PROT-domänen till en fil i OEM-format används ett program skapat i programmeringsspråket C++ (Lippman, 1993). Detta program skapas av författaren till rapporten. Diskussionen rörande C++-koden i detta kapitel kommer att handla om grundläggande tankesätt vid skapandet av koden. Läsare som vill ta del av den fullständiga koden hänvisas till att ta kontakt med författaren av rapporten.

Det huvudsakliga arbetet med C++-koden ligger i att göra en inläsning av de två bokstäver som identifierar vilket innehåll en rad i SWISS-PROT-domänen innehåller. Dessa två bokstäver kommer i fortsättningen att kallas för radens radbeteckning. Utifrån denna inläsning görs sedan en tolkning beroende på radbeteckningens innebörd. Exempelvis måste en åtskillnad göras mellan de två radbeteckningarna ”AC” och ”CC”. Information lagrat under ”AC” är nämligen proteinets unika identifierare medan information lagrat under ”CC” är kommentarer om proteinet.

Efter detta görs en inläsning av radinformationen fram till radslutet. Denna information läggs sedan under ett subobjekt med namnet som radbeteckningen har. I de fall raden börjar på bokstäverna ”CC” eller ”SQ” görs dock en åtskillnad i förfarandet. All information på de rader som börjar på ”CC” läggs nämligen under ett objekt benämnt ”CC”, istället för som i de andra fallen under flera objekt med samma benämning. Om raden börjar på ”SQ” läggs informationen på den första raden under ett objekt ”SQINFO” och resten av informationen på raderna läggs under ett objekt ”SQ”. Den senare innehåller då hela proteinsekvensen beskriven i form av aminosyror, medan ”SQINFO” endast innehåller information om proteinsekvensen (exempelvis längden på proteinsekvensen i form av antalet aminosyror).

6.3.2 Lagring av OEM-fil i databasen

Som sista del ska den skapade OEM-filen lagras i databasen i Lore. Genom ett textuellt gränssnitt och kommandon som är specifika för databashanteringssystemet Lore kan detta göras på ett enkelt sätt. Den version av Lore som används är version 3.0 för SUN OS, single user.

Det första steget är att skapa en databas där man kan lagra proteinerna. Version 3.0 av databashanteringssystemet Lore måste ligga lagrat i den beskrivna katalogen LoreDBHSVer3_0 för att detta kommando ska fungera. I exempel 13 nedan visas

användarens text och den text som genereras av Lore om skapandet av databasen är lyckat.

Exempel 13. Skapande av databasen ProteinDatabas

```
./LoreDBHSVer3_0 > dbcreate ProteinDatabas
Database ProteinDatabas created successfully
```

Databasens namn är ”ProteinDatabas”. Databashanteringssystemet Lore har nu skapat plats för lagring av data i databasen ”ProteinDatabas”.

Nästa steg är att lagra OEM-filen i databasen. För att detta ska fungera måste filen lagrats i samma bibliotek som databashanteringssystemet, det vill säga LoreDBHSVer3_0. I exempel 14 nedan visas användarens text för att lagra OEM-filen i databasen tillsammans med den text som genereras av Lore om lagringen av OEM-filen är lyckad.

Exempel 14. Lagring av OEM-fil i databasen ProteinDatabas

```
./LoreDBHSVer3_0 > dbload2 ProteinDatabas Protein.oem
Registering namn: ProteinDB
File Protein.oem loaded successfully
```

Detta är de steg som behöver tas för att lagra data i databasen. För att testa att databasen verkligen existerar kan man provköra databasen med kommandosträngen som visas i exempel 15. Efter uttrycket ”Lore:” skrivs de frågor användaren vill ställa mot databasen.

Exempel 15. Teststart av databasen ProteinDatabas

```
./LoreDBHSVer3_0 > lore ProteinDatabas
Lore:
```

Samma förfarande som beskrivs i exempel 15 kommer att användas för att göra de testningar som beskrivs i nästa kapitel. Avslutningsvis ska det sägas att ungefär 20%, 221 proteiner, av de proteiner som låg lagrade i SWISS-PROT-domänen lagrades i databasen i Lore.

6.4 Resultat

Föregående kapitel tog upp de arbetsmoment som utfördes vid skapandet av databasen. Det resultat som framkom ur detta arbete var just en databas i Lore. I kapitel 6.2.2 visas en modell över databasen. I denna modell visas endast en del av den information som kan lagras i databasen. Nedanstående tabell visar alla de olika entiteter, representerade som objekt, som kan förekomma i databasen. Med detta följer även en beskrivning på vilken information entiterna innehåller.

Radbeteckning i SWISS-PROT-domänen	Objektnamn i databasen i Lore	Förklaring på objektets information
ID	ID	Ett användarvänligt namn på proteinet
AC	AC	Unikt identifierings-nummer på proteinet
DT	DT	Datum då proteinet lagrats eller då modifieringar gjorts
DE	DE	Ytterligare namn på proteinet
GN	GN	Proteinets gen-namn
OS	OS	Proteinets artnamn
OC	OC	Proteinets organism-klassificering
R (X)	R (X)	(X) representerar en bokstav. Informationen i dessa objekt kan exempelvis vara referenser från litteratur
CC	CC	Kommentarer om proteinet som inte kan placeras under något annat objekt
DR	DR	Referenser till andra molekylärdatabaser
KW	KW	Relevanta nyckelord till proteinet
FT	FT	Intressanta regioner i proteinsekvensen och så vidare
SQ	SQINFO SQ	SQINFO innehåller information om sekvensen, antalet aminosyror och så vidare. SQ innehåller hela proteinsekvensen beskriven i aminosyror

Bild 2. Tabell med den information som ligger lagrad i databasen i Lore

Nästa kapitel kommer bland annat att beskriva en testning som genomfördes mot databasen. Denna testning ska se om arbetshypotesen kan verifieras eller falsifieras.

7 Slutsatser

I föregående kapitel beskrevs resultatet av det arbete som utförts. Detta kapitel tar upp en analys av resultatet, i form av en testning mot databasen, samt slutsatser som kan dras från det genomförda arbetet.

7.1 Testning av databasen

När lagringen av data gjorts kan det vara intressant att kontrollera om de krav som ställdes på databasen i början av genomförandet har blivit uppfyllda. Detta kan man göra genom en testning mot databasen. I större projekt kan en simulator användas för att säkrare fastställa om programvaran är korrekt och utan fel. Detta projekt kommer dock endast att använda en testning mot databasen och inte en simulator, eftersom det är ett mindre projekt. En motivering till att inte använda sig av en simulator i detta projekt är att denna troligtvis skulle ta lika lång tid (om inte längre) att skapa som skapandet av databasen tog. Tid och resurser till att skapa denna simulator finns inte.

Tester mot databasen kan göra att fel upptäcks som skaparen av databasen i efterhand helt eller delvis kan rätta till. Det finns naturligtvis oändliga testningsmöjligheter för att kontrollera databasen. Tid och resurser är dock de faktorer som sätter begränsningarna i detta arbete för vilket antal och vilka tester som kommer att utföras.

- Tester som ger resultat innehållande liknande information från alla proteiner lagrade i databasen. Detta resultat kallas i rubriken i kapitel 6.4.1 för ett täckande resultat.

De tester som valts för detta arbete är:

- Tester som ger resultat innehållande unik information från endast ett protein lagrat i databasen.
- Tester som använder generella sökuttryck, exempelvis "**%**" och "**#**".

7.1.1 Testning med täckande resultat

Testerna beskrivna i detta kapitel skall ge svar på om det går att få ut liknande information från alla proteiner lagrade i databasen. I exempel 16 ställs en fråga mot databasen som skall generera ett resultat med alla proteiners unika identifierare. Denna information ligger under objektet "AC" i databasen. Frågan och resultatet visas nedan.

Exempel 16. Fråga som genererar alla unika identifierare för alla proteiner lagrade i databasen

```
Select ProteinDB.ProteinInfo.AC;
Answer
AC P23279
AC P16536
AC Q06402
AC P30496
AC P16624
      .
      .
      .
```

```
AC P30450
AC P46729
AC P30504
AC P16909
AC P01891 P10315
```

I kapitel 6.3 räknades det antal proteiner lagrats i databasen. Resultatet gav 221 proteiner. Kontrollen av de förekomster av unika identifierare som genererades i resultatet i exempel 16 ger också antalet 221.

I vissa fall motsvarar dock inte den täckande sökningen det sammanlagda antalet lagrade proteinsekvenser i databasen. Detta beror på att informationen saknas. I exempel 17 visas en fråga som skall generera ett resultat med alla proteiners gennamn.

Exempel 17. Fråga som genererar alla gen-namn för alla proteiner lagrade i databasen

```
Select ProteinDB.ProteinInfo.GN;
Answer
GN HLA-A OR HLAA.
GN OMP.
GN HLA-C OR HLAC.
GN AFT1.
GN HLA-B OR HLAB.
      .
      .
      .
GN OMP.
GN HLA-B OR HLAB.
GN ACC1.
GN HLA-B OR HLAB.
GN HLA-B OR HLAB.
```

I exempel 17 kontrollräknas inte förekomsterna av gen-namn för att se om dessa motsvarar antalet lagrade proteiner i databasen. Detta beror på att flera gen-namn kan tillhöra samma protein, vilket har sin grund i överföringen av data från SWISS-PROT-domänen till en fil i OEM-format. För att visa att ett protein kan ha flera gennamn visas i exempel 18 en fråga som genererar de gen-namn proteinet med den unika identifieraren "Q62671" har.

Exempel 18. Fråga som genererar alla gen-namn för proteinet med den unika identifieraren "Q62671"

```
Select ProteinDB.ProteinInfo.GN
Where ProteinDB.ProteinInfo.AC = "Q62671";
Answer
```

Med exempel 16 har vi visat en sökning mot databasen för att finna alla unika identifierare för alla lagrade proteiner. Resultatet gav det exakta antal förekomster av

unika identifierare som ligger lagrat i databasen. Detta kunde bevisas med hjälp av den räkning som gjorts av antalet proteiner i kapitel 6.3

Vi visade med exempel 17 att en sökning för att finna information från alla proteiner ibland inte gav önskat resultat. Vissa proteiner har inte denna information, eftersom den inte existerar för detta protein. Detta visades med exempel 18 för proteinet med den unika identifieraren Q62671.

7.1.2 Testning med unika resultat

Testerna beskrivna i detta kapitel skall ge svar på om det går att få ut unik information från endast ett protein lagrat i databasen. I exempel 19 ställs en fråga mot databasen som skall generera ett resultat med ett proteins korsreferenser till andra databaser. Denna information ligger under objektet "DR" i databasen. Proteinet har den unika identifieraren "P49106". All information om proteinet beskrivs i detalj i exempel 1 i kapitel 2.2.1. Bild 2 i kapitel 6.2.2 visar dessutom en modell över databasen med en del av proteinets information.

Exempel 19. Fråga som genererar ett proteins korsreferenser till andra databaser. All information om proteinet visas i exempel 1 i kapitel 2.2.1.

```
Select ProteinDB.ProteinInfo.DR
Where ProteinDB.ProteinInfo.AC = "P49106";
Answer
DR EMBL S77133 G998430 -.
DR MAIZEDB 65832 -.
```

Vid sökningar för att finna unik information kan det visa sig att flera svar genereras. Detta visades delvis i exempel 17. I exempel 20 ställs en fråga mot databasen som skall generera ett resultat innehållande datum specifika för ett protein. Datumen beskriver exempelvis när proteinet lagrades i databasen och (om information existerar) när och hur proteinet modifierats. Denna information ligger under objektet "DT" i databasen. Proteinet som sökningen utförs mot är detsamma som beskrevs i exempel 19. I resultatet av frågan ser vi att flera förekomster av datum förekommer för detta protein.

Exempel 20. Fråga som genererar datum specifika för ett protein.

```
Select ProteinDB.ProteinInfo.DT
Where ProteinDB.ProteinInfo.AC = "P49106";
Answer
DT 01-FEB-1996 (REL. 33, CREATED)
DT 01-FEB-1996 (REL. 33, LAST SEQUENCE UPDATE)
DT 01-OCT-1996 (REL. 34, LAST ANNOTATION UPDATE)
```

Med exempel 19 har vi visat en sökning mot databasen och det svar som genererades för att finna ett unikt proteins korsreferenser till andra databaser. Detta resultat visade bara en förekomst av en korsreferens. För att visa att en unik sökning också kan generera flera förekomster visades i exempel 20 en fråga som resulterar i de datum som är specifika för ett unikt protein.

7.1.3 Testning med generella sökuttryck

I kapitel 3.2.2 beskrevs användandet av generella sökuttryck. Dessa skulle bland annat underlätta sökandet av information för användaren av databasen. I detta kapitel

beskrivs några exempel med generella sökuttryck mot den skapade databasen. Det första exemplet använder sig av uttrycket "**%**" för att matcha noll eller flera bokstäver hos ett objekt i modellen av databasen. Anta att användaren av databasen vill ta reda på organismklassificeringen för proteinet som beskrivs i texten ovanför exempel 19. Denna information ligger under objektet "OC" i databasen. Användaren känner till proteinets unika identifierare, men är tveksam till hela vägbeskrivningen till objektet "AC". Han känner till ingångsnamnet till databasen ("ProteinDB"), men inte hela namnet på objektet mellan "ProteinDB" och "AC". Det enda han vet är att detta objekt börjar på ordet "Protein". I detta fall kan sökuttrycket "**%**" användas för att matcha resten av sekvensen efter ordet "Protein". Detta visas som en fråga och ett resultat i exempel 21.

Exempel 21. Fråga som genererar ett proteins organismklassificering

```
Select ProteinDB.Protein%.OC

Where ProteinDB.Protein%.AC = "P49106";

Answer

   OC EUKARYOTA PLANTA EMBRYOPHYTA ANGIOSPERMAE
   MONOCOTYLEDONEAE

   OC CYPERALES GRAMINEAE.
```

Det generella sökuttrycket "**#**" motsvarar till stor del "**%**". Skillnaden är att "**#**" matchar ett helt objektnamn, vilket kan vara användbart i nästa exempel där vi använder oss av "*path-of*"-funktionen. Denna funktion tar fram en väg till ett visst objekt i modellen. Anta att användaren av databasen inte känner till databasens struktur och att han vill ta fram alla unika identifierare för proteiner lagrade i databasen. För att kunna söka efter dessa måste han dock ha den exakta vägen till objektet som lagrar den unika identifieraren. Som i föregående exempel känner han till ingångsnamnet till databasen, "ProteinDB", och "AC". Vilka objekt som ligger där emellan vet han däremot inte. Sökvariabeln P ger i exempel 22 den exakta vägen mellan "ProteinDB" och "AC". Användaren vet dock inte att det i detta fall endast är objektnamnet "ProteinInfo", som ges som resultat.

Exempel 22. Fråga som genererar vägen till en unik identifierare i databasen

```
Select distinct path-of (P)

From ProteinDB.#@P.AC

Answer

   ProteinInfo
```

Med exempel 21 har vi visat en sökning mot databasen för att finna ett unikt proteins organismklassificering. Härvid använde vi oss av det generella sökuttrycket "**%**". I exempel 22 använde vi oss av det generella sökuttrycket "**#**" och "*path-of*"-funktionen för att låta en användare finna en specifik väg till viss information lagrad i databasen. Resultatet gavs som de objektnamn som matchas mot variabeln P.

7.2 Slutsatser

I denna rapport har arbetet med att ta fram en biodatabas i Lore beskrivits. Ur detta arbete kan man nu dra ett antal slutsatser. Dessa slutsatser beskrivs i punktform nedan.

- Arbetet i denna rapport bygger på en arbetshypotes som säger att det går att skapa en biodatabas i Lore. En verifiering av arbetshypotesen kunde endast utföras om alla krav upptagna i kravanalysen uppfylldes. Genom en testning mot den framtagna biodatabasen uppfylldes dessa krav. Därmed verifierades arbetshypotesen. Slutsatsen av detta arbete, enligt denna metod, är att det gick att skapa en biodatabas i Lore.
- Svar från frågor i frågespråket Lorel som ställdes mot biodatabasen genererades snabbt och gav den information som motsvarade den ställda frågan. En positiv aspekt var att objektnamnet skrevs ut innan informationen som låg lagrat i objektet. Detta gjorde resultatet enklare att överblicka.
- Användaren av biodatabasen kan söka efter information om ett specifikt protein i biodatabasen och få ett korrekt svar. Den unika information som beskriver ett protein kan genereras som ett resultat av en sökfråga mot biodatabasen om användaren vet var denna information ligger lagrad.
- Användaren av biodatabasen kan söka efter information rörande en mängd proteiner i biodatabasen och få ett korrekt svar.

Det ska dock kommenteras att alla de kombinationer av frågor som är möjliga att ställa mot biodatabasen inte har testats. Detta är en omöjlig uppgift med tanke på den tid som skulle behöva användas för detta ändamål. De svar som genereras från sökfrågor mot biodatabasen kan däremot jämföras med den information som ligger lagrad i SWISS_PROT-domänen eller filen i OEM-format. Detta innebär att endast de exempel på sökfrågor som visats i kapitel 6.4 representerar dessa påstående som beskrivs i punktform ovan.

8 Diskussion

Detta kapitel kommer att ta upp diskussionsämnen kring arbetet utfört i rapporten. I nedanstående kapitel beskrivs detta följt av ett kapitel med framtida arbeten.

8.1 Arbetet i sammandrag

Det första arbetsmoment som genomfördes i rapporten var en litteraturstudie av de områden som skulle användas i senare del av rapporten. I detta fall var det molekylärbiologiområdet med inriktning på biodatabasen SWISS- PROT (Apweiler och Bairoch, 1998) samt databasområdet med inriktning på databashanteringssystemet Lore (Abiteboul m.fl., 1997b). Denna del av rapporten trodde jag skulle vara ganska enkel att genomföra med tanke på det intresse som fanns för båda dessa områden. I fallet med SWISS-PROT stämde detta ganska väl. Med Lore visade det sig dock att denna studie skulle bli ganska omfattande, eftersom det inte räckte med att bara känna till dess bakgrund. Det krävdes dessutom en förståelse för hur användaren rent praktiskt, genom frågespråket Lorel (Abiteboul m.fl., 1997c), skulle gå tillväga för att få åtkomst till lagrad data i en databas. Detta berodde på att testfrågor i Lorel skulle ställas mot databasen i senare del av rapporten. Det kan dock sägas att den tid som lades ner på studien av SWISS-PROT och Lore i början av arbetet troligen gjorde att mindre tid behövde läggas på de olika faserna vid skapandet av databasen. Databasskapandet kunde på detta sätt till större del fokuseras på designen av databasen och till mindre del på det konkreta skapandet av databasen.

Denna betraktelse gav den största erfarenheten i detta arbete. Genom studien av Lore finns det dessutom möjligheter att använda databashanteringssystemet i andra arbeten, då grunden för hur man använder systemet redan är lagt.

8.2 Framtida arbeten

Författaren till arbetet ser många utvecklingsmöjligheter inom området databaser och molekylärbiologidata. I detta arbete har fokus lagts på databashanteringssystemet Lore (Abiteoul m.fl., 1997b) och lagring av proteiner från en SWISS-PROT-domän. Genom att använda ett annat databashanteringssystem och en annan typ av molekylärbiologidata kan man komma fram till andra resultat och slutsatser.

SWISS-PROT (Apweiler och Bairoch, 1998) är en av många biodatabaser som lagrar molekylärbiologidata. Denna biodatabas lagrar dessutom endast proteiner av primär struktur. Andra biodatabaser lagrar andra strukturer på proteiner. Detta kan exempelvis vara proteiner beskrivna i 3D-struktur.

De framtida utvecklingsmöjligheter som finns med arbetet beskrivet i denna rapport beskrivs i punktform nedan.

- Ett arbete skulle kunna vara att titta på uppdelningen av underkategorier hos ett protein. I SWISS-PROT-domänen representeras exempelvis proteinet beskrivet i aminosyror under beteckningen "SQ". Det är dock inte bara denna information som ligger lagrat under "SQ". Information som exempelvis hur många aminosyror som ingår i proteinet ligger också lagrat under denna beteckning. I den skapade biodatabasen har detta uppmärksammats och proteinet beskrivet i aminosyror ligger då under underkategorin "SQ" medan övrig information ligger under underkategorin "SQINFO". Det finns dock information om ett protein som skulle kunna underkategoriseras ytterligare för

att underlätta och förenkla sökandet i databasen för användaren. Ett exempel på detta är kommentarerna om ett protein som ligger lagrade under beteckningen "CC".

- Ett arbete skulle kunna vara att skapa ett sökverktyg mot biodatabasen för att förenkla sökandet för användaren. Med detta sökverktyg skulle matchningar mellan olika proteiner och mellan olika protein-arter kunna göras, men också sökningar för att finna information om ett specifikt protein

8.3 Visioner

Med arbetet och resultatet som framkommit i rapporten ser författaren de möjligheter som finns med en kombination av molekylärbiologidataområdet och databasområdet. Genom att endast ha mindre kunskaper inom ett frågespråk kan väldigt preciserad information sökas och ett lättöverskådligt resultat genereras. Detta har beskrivits i denna rapport.

Molekylärbiologiområdet beskrevs tidigt i rapporten som ett område som expanderat under senare tid med större mängder utforskad data som resultat. Författarens vision är att denna data ska lagras i biodatabaser liknande denna skapad i detta arbete. Genom länkning ska det sedan vara möjligt för användaren att på ett enkelt sätt få tillgång till unik information fast med ett sökfält mycket större än det i biodatabasen i Lore.

Referenser

S. Abiteboul, R. Goldman, K. Haas, Q. Luo, J. McHugh, S. Nestorov, D. Quass, A. Rajaraman, H. Rivero, J. Ullman, J. Widom och J. Wiener. LORE: A Lightweight Object REpostiory for Semistructured Data. I *Proceedings of the ACM SIGMOD International Conference on Management of Data*, sidan 549, Montreal, Kanada, Juni 1996. Demonstrationsbeskrivning.

S. Abiteboul. Querying semistructured data. I *Proceedings of the International Conference on Database Theory*, sidorna 1-18, Delfi, Grekland, Januari 1997a.

S. Abiteboul, R. Goldman, J. McHugh, D. Quass och J. Widom. Lore: A database management system for semistructured data. I *SIGMOD Record*, 26 (3): 54-66, September 1997b.

S. Abiteboul, J. McHugh, D. Quass, J. Widom och J. Wiener. The Lorel Query Language for Semistructured Data. I *Journal of Digital Libraries*, 1 (1):68-88, April 1997c.

R. Apweiler och A. Bairoch. The SWISS-PROT protein sequence data bank and its supplement TrEMBL in 1998. I *Nucleic Acids Research*, 26 (1): 38-42, 1998.

T.K. Attwood och D.J. Parry-Smith. Introduction to bioinformatics. Addison Wesley Longman, London, England, 1999.

P. Buneman. Semistructured Data. I *Proceedings of the Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, sidorna 117-121, Tucson, Arizona, Maj 1997. Tutorial.

R.G.G. Catell. *The Object Database Standard: ODMG-93*. Morgan Kaufmann, San Francisco, Kalifornien, 1994.

R. Elmasri, S. B. Navathe. *Fundamentals of database systems*. Addison-Wesley, 2000.

European Bioinformatics Institute och Swiss Institute for Bioinformatics, SWISS-PROT. Söksida tillgänglig på Internetadressen: http://www.expasy.ch/sprot/, 2001.

H. Garcia-Molina, Y. Papakonstantinou och J. Widom. Object exchange across heterogenous information sources. I *Proceedings of the Eleventh International Conference on Data Engineering*, sidorna 251-260, Taipei, Taiwan, Mars 1995.

R. Goldman, J. McHugh och J. Widom. From Semistructured Data to XML: Migrating the Lore Data Model and Query Language. I *Proceedings of the 2nd International Workshop on the Web and Databases* (WebDB ´99), Philadelphia, Pennsylvania, Juni 1999.

R. Goldman och J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. I *Proceedings of the TwentyThird International Conference on Very Large Data Bases*, Aten, Grekland, Augusti 1997.

D. B. Lenat. I *Communications of the ACM*, volym 38, November 1995.

S. B. Lippman. *C++ Primer 2nd Edition*. AT & T Bell Laboratories, 1993.

K. Mahesh och S. Nirenburg. I *Proceedings of the FLAIRS-96 Track on Information Interchange, Florida AI Research Symposium*, Maj 1996.

J. McHugh och J. Widom. Integrating dynamically-fetched external information into a dbms for semistructured data. I *Proceedings of the Workshop on Management of Semistructured Data*, sidorna 75-82, Tucson, Arizona, Maj 1997.

S. Schulze-Kremer. Ontologies for molecular biology. I *Proceedings of the Third Pacific Symposium on Biocomputing, World Scientific Publishers*, sidorna 693-704, Singapore, 1998.

U. S. Department of Energy, Office of Energy Research, Office of Biological and Environmental Research. 1997 Human Genome Program Report. November 1997.