SISU analys

Nr. 5

DATABASER - enkla att använda

HSQL,
ett kunskapsbaserat hjälpsystem för SQL

Erik Knudsen

Redaktör: B. Anders Eriksson
SISU - Svenska Institutet för systemutveckling

Box 1250
163 13 SPÅNGA

Kistagången 26
KISTA

08- 750 7500

DATABASER - enkla att använda

HSQL, ett kunskapsbaserat hjälpsystem för SQL

Erik Knudsen

Redaktör: B. Anders Eriksson

ISSN: 0282-9924

Historien upprepar sig?

Xian i Kina är framför allt känt för terrakottafigurerna som kejsare Qin lät ställa att vakta hans grav. Mindre känt är att där finns även ett provinsmuseum som bl a har i sina samlingar några av världens tyngsta böcker. De är gjorda av stenblock och antalet block är c:a 2.300. Hur de hittar bland de upp till 2.000 år gamla "boksidorna" vet jag ej, men att göra ett rispappersavtryck av en sten tar från en till ett par timmar. Och det görs av specialister för att det skall göras på ett riktigt sätt.

Smidigare "böcker" använder vi idag, men de blir allt fler och allt mer data samlar vi på oss, inte bara i form som böcker. Problemet idag är att hitta bland all data som finns lagrat. Trenden har varit att mer och mer avancerade specialister har gjort sökningar efter data och det har tagit lång tid.

Det här numret av SISU-analys behandlar vilka vägar det finns att underlätta sökning i databaser där SQL går att använda. Artikeln är skriven av Erik Knudsen och han är systemvetare, utbildad vid Institutionen för ADB, Stockholms universitet. Därefter arbetade han med utveckling av kunskapsbaserade system på Infologics. Idag är han konsult på SISU och kombinerar det med att undervisa i logikprogrammering och att forska i naturliga språk hos SYSLAB.

Om databaserna inte skall gå samma öde till mötes som "stenböckerna" i Xian så måste det bli lättare att nå data som är lagrat i alla våra databaser.

B Anders Eriksson

Innehåll

Inledning

I takt med att allt fler människor som inte är dataexperter kommer i kontakt med datorsystem krävs en alltmer raffinerad funktionalitet hos de gränssnitt som finns hos systemen. Samtidigt som användarna blir fler och mer heterogen som grupp ställs också krav från dessa grupper på bättre interaktivitet mellan de själva och systemen. Detta gäller i allmänhet för dialogsystem men i synnerhet för informationssystem.

Huvudsyftet med artikeln är att göra en grov specifikation av ett kunskapsbaserat hjälpsystem för att stödja användare som använder SQL som frågespråk för informationsåtervinning - Hjälpsystem för SQL (HSQL) - samt att till en viss del ange vilka konsekvenser det för med sig att realisera det. Vidare föreslås ett antal tänkbara ansatser som kan ligga till grund för en implementering av ett valt gränssnitt.

Denna artikel skall som framgår av syftet kunna utgöra underlag för en mer detaljerad systemspecificering. Dock är det vår avsikt att presentera materialet på ett sådant sätt att det även kan läsas och förstås av icke-datorexperter. Detta för att även de som slutligen kommer i kontakt med ett färdigt system skall kunna bilda sig en klar uppfattning om vad det innebär att använda ett kunskapsbaserat hjälpsystem för SQL. Till detta kommer de som skall stå för en eventuell utveckling av HSQL.

Avsnitt 1 ger en allmän introduktion till vissa centrala begrepp som databaser, databashanteringssystem etc. En beskrivning av de problem som är förknippade med användning av SQL som frågespråk görs.

Avsnitt 2 beskriver endast den frågande delen av SQL. Detta görs enligt principen förklara genom att exemplifiera.

Avsnitt 3 ger en beskrivning av några olika typer av gränssnitt som kan anses vara grundläggande. De utmärkande dragen presenteras och exempel på användningsområden ges. En diskussion förs kring varje gränssnitt i avsikt att påvisa när respektive gränssnitt är lämpligt eller olämpligt.

I avsnitt 4 specificeras de funktioner som skall ingå i ett kunskapsbaserat system för att stödja användningen av SQL som frågespråk. Den diskussion som förs är dock generell och giltig för de flesta typer av system för informationsåtervinning. Vidare ges en tänkbar logisk konfiguration i avsikt att klart separera de urskiljbara beståndsdelarna.

I avsnitt 5 beskrivs fem olika typer av ansatser som var och en utgör en möjlig kandidat som gränssnitt i ett kunskapsbaserat hjälpsystem.

I och med att avsnitt 2-3 är av förklarande och introducerande karaktär kan de läsare med kännedom om grundläggande begrepp inom ADB och informationssystem gå direkt till avsnitt 4.

Vi vill samtidigt tacka Janis Bubenko, Christer Dahlgren, Eva- Louise Eliasson, Gregor Jonsson, Eva Lindencrona-Ohlin samt Björn Nilsson för deras värdefulla synpunkter, kommentarer och råd om innehållet i denna artikel. Stort tack även till Marianne Sindler för hennes insats med att färdigställa densamma.

1. Databaser och deras användare

Alltmer information övergår från manuell till datoriserad bearbetning. Information som tidigare fanns nedtecknad på papper finns nu lagrad i digital form på något lämpligt media, tex magnetband eller skivminnen. Datorisering har inneburit att smått fantastiska mängder data numera kan lagras på en bråkdel av det utrymme som tidigare krävdes för motsvarande informationsmängd. Samtidigt kan vi få en mycket snabb tillgång till dessa data och sammanställa dessa alltefter de informationsbehov som finns.

Denna glättade framställning brukar ofta presenteras och användas som argument för införande av datoriserade system. Den verkliga bilden kanske inte alltid är så ljus. Det är otvivelaktigt så att datorer kan lagra enorma mängder data och utföra räkneoperationer med en hastighet så tanken svindlar. Dock är det inte fallet att det alltid går snabbt och lätt att få tillgång till lagrade data.

Innan den problematiken studeras närmare behandlas grundläggande begrepp för databaser.

Vad är databaser?

En databas är såsom namnet antyder en samling av data. För att det skall vara möjligt att veta var olika data finns måste det finnas en beskrivning av de data som är lagrade. Beskrivningen skall ange hur data hänger ihop. Vidare behövs en beskrivning av vad data betyder, dvs hur skall varje enskilt datum tolkas. Det brukar kallas för en logisk beskrivning av databasens innehåll, "logiskt schema" är den gängse terminologin, (eng data dictionary).

Observera att vi hittills enbart har talat om data, inte om information. Enbart råa data ger knappast någon information. Det är först när en tolkning av data finns tillhanda som man kan tala om information. Enkelt uttryckt: information = data + tolkning. Det logiska schemat hjälper oss således att tolka databasens innehåll.

Resonemanget kan exemplifieras på följande sätt: Någonstans i databasen finns följande data, 860101, lagrat. Vad betyder det? Är det månne ett datum eller vad, och om det är ett datum, är det ett födelsedatum? Om vi ser efter i vårt logiska schema finns där en uppgift om att detta datum anger antalet aktier som en viss person äger.

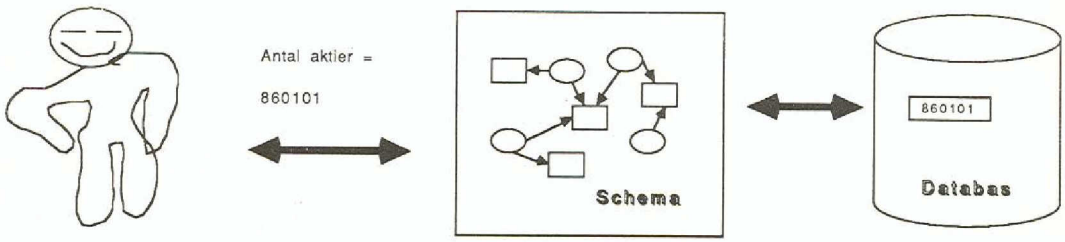


Fig 1.1 Schemat ger korrekt tolkning

I dagsläget sker inte denna tolkning enbart med det logiska schemat. Det är ofta fallet att tolkningen av data även finns i de aktuella applikationsprogrammen.

Ett informationssystem (IS) kan således bestå av en databas med beskrivningar av databasens innehåll samt som regel en mängd olika program som möjliggör kommunikation mellan en användare av ett IS och informationssystemet självt.

Med ett databashanteringssystem, DBHS, menas de generella, ej applikationsberoende, program som används för att administrera och kommunicera med databasen.[1] Enklast kan man betrakta ett DBHS som ett programsystem som kan användas i en mängd olika tillämpningar.

Det existerar ett antal olika typer av DBHS där den grundläggande skillnaden mellan dessa är principen för hur data logiskt skall organiseras. De tre idag mest utbredda typerna är hierarkiska-, nätverk- och relationsdatabaser. Man brukar i stället för typer tala om datamodeller. Modellen beskriver principen för en logisk organisering.

Det finns en mängd olika DBHS som bygger på någon av de ovan nämnda modellerna. Några exempel: IMS, DMS-1100, DB2.

DBHS som bygger på de två förstnämnda modellerna är väl beprövade och har funnits i drift länge, för att vara i datorsammanhang. Utan att närmare studera dessa kan sägas att de datamodeller som de baseras på inte direkt stödjer ett dynamiskt informationsbehov. Organisationer, som har ett känt informationsbehov och där detta behov i stort sett aldrig förändras eller förändras långsamt över tiden, kan säkerligen klara sig med vilken typ av DBHS som helst. Det blir andra kriterier som fäller avgörandet för val av DBHS. Organisationer däremot, med informationsbehov som förändras i snabb takt och där viss information efterfrågas sporadiskt eller blott ett fåtal gånger, bör välja DBHS med omsorg.

Tidigare var situationen den att när en användare av ett informationssystem ville få ut en ny typ av information eller till viss del förändrad information mot normalt krävdes det att en programmerare skrev ett nytt, alternativt ändrade i ett befintligt, program för att tillgodose de nya behoven. Detta är en process som kan ta alltför lång tid i anspråk särskilt i en tidskritisk miljö. Tanken är således att användare av ett IS själva skall kunna kommunicera med systemet direkt utan att gå omvägen via en programmerare.

Numera kan man genom att välja ett lämpligt DBHS, tex ett som bygger på relationsmodellen och som har ett härför avsett språk, kommunicera direkt med ett informationssystem. Ett sådant språk är SQL (Structured Query Language). Detta beskrivs i avsnitt 2. Även om vi i denna artikel enbart berör SQL finns frågespråk för andra typer av DBHS, tex QLP (Query Language Processor) som används i DBHS baserat på nätverks-modellen.

Tanken är således att göra informationen mer lättillgänglig och lättåtkomlig genom att använda sk frågespråk av typen SQL eller liknande. Dessa språk skall användare av ett IS utnyttja för att tillgodose sitt eget informationsbehov.

---

[1] (Da81) Alla referenser anges i fortsättningen med fotnötter.

Användare av databaser

Med användare av ett IS avses de personer som tidigare fick information genom kartotek, liggare, böcker etc, men som nu skall kunna få motsvarande information genom ett datoriserat system. Vi avser inte personer vilkas befattningar uppstår som ett resultat av ett IS tillkomst, dvs systemadministratörer, programmerare osv.

Användare av ett IS kan grovt delas in i två kategorier.

Den första kategorin utgörs av användare som nyttjar systemet med en sådan frekvens att de med tiden erhåller en klar bild av databasens innehåll. Vidare utgör frågespråkets syntax inget hinder då den höga användningsfrekvensen gör att dessa snabbt utökar sin kunskap i att formulera alltmer komplexa frågor. Med andra ord, dessa användare kan relativt lätt formulera sina utsökningskrav i SQL och erhålla de svar som de förväntar sig.[2]

Den andra kategorin utgörs av användare som nyttjar informationssystemet mer sporadiskt. De har en oklar eller ytlig bild av databasens innehåll. Eftersom de nyttjar systemet mer sällan blir SQL-syntaxen ett svårframkomligt hinder då kunskapen om hur man formulerar frågor hinner falla i glömska mellan användningstillfällena.

Problemen med SQL gäller således huvudsakligen den andra kategorin av informationssystemanvändare. Dessa behöver ett betydligt bättre stöd av systemet än vad som kan erhållas idag. Ett sådant stöd realiseras lämpligen med hjälp av kunskapsteknik. Denna teknik beskrivs närmare i avsnitt 4.

2. FRÅGESPRÅK

Avsnittet ger en introduktion till begreppet relationsdatabas samt beskriver ett språk för datamanipulering, SQL (Structured Query Language). Beskrivningen av detta och givna exempel bygger till stor del på material ur Dates bok, An introduktion to Database Systems.[3]

Relationsdatabas

Tidigare har nämnts att det finns ett antal olika typer av DBHS varav relationsdatabaser (RDBHS) verkar vara en av de idag mest lovande typerna för snabb åtkomst av information i ett IS där informationsbehovet är dynamiskt. Ett skäl till att relationsdatabashanteringssystem blivit tämligen populära är det faktum att det är tämligen enkelt att få en överblick över databasen samt enkelt kunna manipulera densamma. Eftersom SQL är ett språk som utgår från RDBHS kan det vara på sin plats att ge en kortfattad beskrivning av hur den typen av databaser ser ut och fungerar.

En relationsdatabas kan sägas vara en samling av tabeller. I dessa tabeller lagras enskilda data. Varje tabell har ett namn som identifierar tabellen. Vidare har varje

[2] (Re81)
[3] (Da81)

tabell en eller flera kolumner som också namnges.

Ett exempel på en tabell över telefonabonnenter skulle kunna vara följande:

Tabell: Abonnent

Tfn	Namn	Adress	Pnr	Padr
123456	Sven Svensson	Storgatan 3	12345	Storstad
156789	Olga Olsson	Lillgatan 8	14545	Storstad
238562	Sven Svensson	Bogatan 23	34567	Småstad
456735	Nils Nilsson	Laxgatan 34	76523	Fiskeby

Fig 2.1 Exempel på tabell i relationsdatabas

I kolumnen tfn lagras endast telefonnummer, i namnkolumnen endast för och efternamn osv. På frågan om vem som har telefonnummret 156789 letar vi upp den rad i tabellen som har samma nummer och ser i kolumnen för namn att det är Olga Olsson.

156789 Olga Olsson Lillgatan 8 14545 Storstad

På frågan om vilket telefonnummer som Sven Svensson har letar vi på motsvarande sätt och erhåller två nummer: 123456 och 238562.

123456 Sven Svensson Storgatan 3 12345 Storstad
238562 Sven Svensson Bogatan 23 34567 Småstad

Om vi vet att det är Sven Svensson på Storgatan 3 kan vi genom att jämföra både namn- och adresskolumnen erhålla rätt telefonnummer.

123456 Sven Svensson Storgatan 3 12345 Storstad

Detta är mycket förenklat den princip för hur relationsdatabaser är uppbyggda och delvis fungerar.

Notera att exemplet endast avsåg utsökning ur en tabell. Givetvis går det att förändra eller uppdatera en tabells innehåll genom att lägga till nya rader, förändra en befintlig rads värden eller helt ta bort en rad. Vidare kan nya tabeller genereras genom att slå samman vissa kolumner i olika tabeller.

För att kommunicera med ett DBHS finns det ett antal olika konstruerade språk som kräver en strikt syntax, strikt i den mening att en användare av språket i alla lägen måste uttrycka sig korrekt.

Strukturerat frågespråk

Structural Query Language (SQL) är ett datamanipuleringsspråk dvs ett språk med vars hjälp en databas innehåll kan förändras.[4] Inom ramen för denna artikel kommer endast den frågande delen av SQL att behandlas. Med frågande menas att det enbart är möjligt att göra utsökningar eller erhålla information från informationssystemet.

Ett par skäl för att välja SQL som datamanipuleringsspråk är dels att det är vida spritt och relativt känt, dels att en standard håller på att utarbetas för SQL av ANSI (American National Standards Institute), som ungefär motsvarar SIS (Standardiseringskommissionen i Sverige).

Även om SQL kan sägas stå högt över andra mer datornära typer av datamanipuleringsspråk är det långt ifrån något naturligt språk för människor. Den som använder SQL tvingas lära sig i princip ett nytt språk, låt vara ett relativt begränsat.

För att kunna uttrycka en för användaren meningsfull fråga i SQL måste denne ha en fullständigt klar bild av det logiska schemat, dvs vilka tabeller som finns, vad de olika kolumnerna heter och vad de innehåller eller betyder etc. Tabellerna i sig ger inget stöd för en semantisk tolkning. Dock skall sägas att SQL är ett kraftfullt språk när man väl behärskar det. Det är vår mening att SQL enklast beskrivs genom exemplifiering. Exempel med stigande svårighetsgrad kommer att presenteras i avsikt att dels beskriva SQL dels visa på styrkan hos SQL men även antyda dess nackdelar när komplexitetsgraden hos en fråga stiger.

---

[4] (IBM83)

Som utgångspunkt för exemplen används följande databas:

Leverantör

LevNr	LevNamn	Status	Stad
L1	Svensson	20	Stockholm
L2	Nilsen	10	Köpenhamn
L3	Hansen	30	Köpenhamn
L4	Olsson	20	Stockholm
L5	Mäki	30	Helsinki

Artikel

ArtNr	Artnamn	Färg	Vikt	Stad
A1	Mutter	Röd	12	Stockholm
A2	Bult	Grön	17	Köpenhamn
A3	Skruv	Blå	17	Oslo
A4	Skruv	Röd	14	Stockholm
A5	Kam	Blå	12	Köpenhamn
A6	Kugge	Röd	19	Stockholm

LevArt

LevNr	ArtNr	Kvantitet
L1	A1	300
L1	A2	200
L1	A3	400
L1	A4	200
L1	A5	100
L1	A6	100
L2	A1	300
L2	A2	400
L3	A2	200
L4	A2	200
L4	A4	300
L4	A5	400

Fig 2.2 Exempel på en relationsdatabas

En tabell med leverantörer där de olika kolumnerna anger leverantörens nummer, namn, status och stad. En annan tabell med artiklar med uppgifter om artikelnummer, namn, färg, vikt och stad. En tredje tabell, LevArt, där varje tabellrad anger ett leverantörsnummer, artikelnummer och en kvantitet.

Observera att LevNr i leverantörstabellen inte behöver betyda samma sak som LevNr i LevArt-tabellen. I stället hade vi kunnat kalla LevNr för Leverantörsnummer. För enkelhetens skull använder vi samma namn, eftersom det är det som avses.

Första raden i LevArt-tabellen kan utläsas:

Leverantör nummer 1 levererar 300 st artiklar med nummer 1.

LevArt-tabellen är den tabell som anger att det råder ett speciellt samband mellan leverantörer och artiklar.

Den grundläggande konstruktionen hos SQL för utsökningar är SELECT..FROM..WHERE.

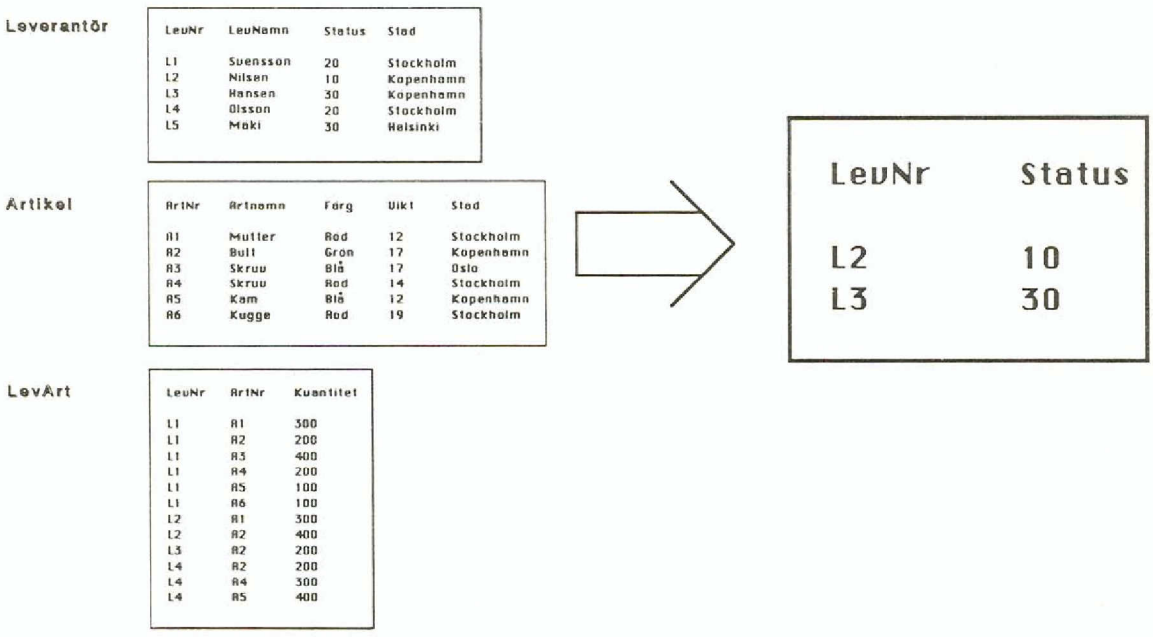
Sökning med testvillkor

Fråga: Lista leverantörsnummer och status för leverantörer i Köpenhamn

```
SELECT   LevNr,Status
FROM     Leverantör
WHERE    Stad='Köpenhamn'
```

Detta är den konstruktion som SQL kräver för att uttrycka ovanstående fråga. Den kan tolkas så att välj ut samtliga rader med kolumner LevNr och Status från leverantörstabellen där stadkolumnen innehåller värdet Köpenhamn.

Resultat:



Sökning med eventuella dubbletter

Fråga: Lista artikelnummer för samtliga artiklar som levereras

```
SELECT  ArtNr
FROM    LevArt
```

Resultat:

Leverantör

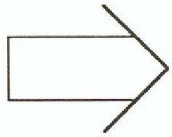
LevNr	LevNamn	Status	Stad
L1	Svensson	20	Stockholm
L2	Nilsen	10	Köpenhamn
L3	Hansen	30	Köpenhamn
L4	Olsson	20	Stockholm
L5	Mäkl	30	Helsinki

Artikel

ArtNr	Artnamn	Färg	Vikt	Stad
A1	Mutter	Röd	12	Stockholm
A2	Bult	Grön	17	Köpenhamn
A3	Skruv	Blå	17	Oslo
A4	Skruv	Röd	14	Stockholm
A5	Kam	Blå	12	Köpenhamn
A6	Kugge	Röd	19	Stockholm

LevArt

LevNr	ArtNr	Kvantitet
L1	A1	300
L1	A2	200
L1	A3	400
L1	A4	200
L1	A5	100
L1	A6	100
L2	A1	300
L2	A2	400
L3	A2	200
L4	A2	200
L4	A4	300
L4	A5	400



ArtNr
A1
A2
A3
.
A1
A2
.

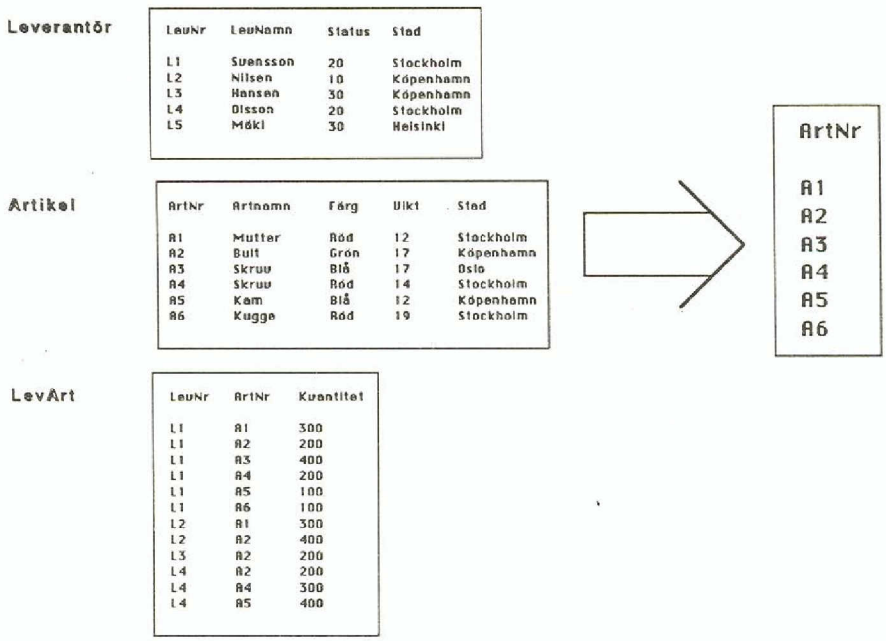
Eftersom SQL inte skönsmässigt tar bort dubbletter finns möjlighet att göra detta med hjälp av UNIQUE, (sv unik).

Sökning utan dubbletter

Fråga: Lista artikelnummer för samtliga artiklar som levereras utan dubbletter

```
SELECT UNIQUE   ArtNr
FROM            LevArt
```

Resultat:



Leverantör

LevNr	LevNamn	Status	Stad
L1	Svensson	20	Stockholm
L2	Nilsen	10	Köpenhamn
L3	Hansen	30	Köpenhamn
L4	Olsson	20	Stockholm
L5	Mäki	30	Helsinki

Artikel

ArtNr	Artnamn	Färg	Vikt	Stad
A1	Mutter	Röd	12	Stockholm
A2	Bult	Grön	17	Köpenhamn
A3	Skruv	Blå	17	Oslo
A4	Skruv	Röd	14	Stockholm
A5	Kam	Blå	12	Köpenhamn
A6	Kugga	Röd	19	Stockholm

ArtNr
A1
A2
A3
A4
A5
A6

LevArt

LevNr	ArtNr	Kvantitet
L1	A1	300
L1	A2	200
L1	A3	400
L1	A4	200
L1	A5	100
L1	A6	100
L2	A1	300
L2	A2	400
L3	A2	200
L4	A2	200
L4	A4	300
L4	A5	400

Visa all data

Fråga: Lista samtliga uppgifter om leverantörer

```
SELECT  *
FROM    Leverantör
```

Resultat: En kopia av leverantörstabellen

Sökning med villkor

Villkorsmässiga utsökningar sker med hjälp av test och jämförelseoperatorer eller logiska operander AND, OR och NOT.

Fråga: Nummer för leverantörer i Köpenhamn med status större än 20

```
SELECT   LevNr
FROM     Leverantör
WHERE    Stad='Köpenhamn'
AND      Status>20
```

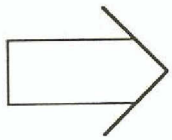
Resultat:

Leverantör

LevNr	LevNamn	Status	Stad
L1	Svensson	20	Stockholm
L2	Nilsen	10	Köpenhamn
L3	Hansen	30	Köpenhamn
L4	Olsson	20	Stockholm
L5	Mäkl	30	Helsinki

Artikel

ArtNr	Artnamn	Färg	Vikt	Stad
A1	Mutter	Röd	12	Stockholm
A2	Bult	Grön	17	Köpenhamn
A3	Skruv	Blå	17	Oslo
A4	Skruv	Röd	14	Stockholm
A5	Kam	Blå	12	Köpenhamn
A6	Kugge	Röd	19	Stockholm



LevNr
L3

LevArt

LevNr	ArtNr	Kvantitet
L1	A1	300
L1	A2	200
L1	A3	400
L1	A4	200
L1	A5	100
L1	A6	100
L2	A1	300
L2	A2	400
L3	A2	200
L4	A2	200
L4	A4	300
L4	A5	400

Sökning med villkor där resultatet är sorterat

Utsökning med resultattabellen sorterad med hjälp av ORDER BY...DESC.

Fråga: Nummer och status för leverantörer i Köpenhamn sorterad i sjunkande ordning över status

```
SELECT   LevNr
FROM     Leverantör
WHERE    Stad='Köpenhamn'
ORDER    BY Status DESC
```

Resultat:

Leverantör

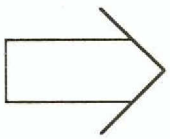
LevNr	LevNamn	Status	Stad
L1	Svensson	20	Stockholm
L2	Nilsen	10	Köpenhamn
L3	Hansen	30	Köpenhamn
L4	Olsson	20	Stockholm
L5	Mäki	30	Helsinki

Artikel

ArtNr	Artnamn	Färg	Vikt	Stad
A1	Mutter	Röd	12	Stockholm
A2	Bult	Grön	17	Köpenhamn
A3	Skruv	Blå	17	Oslo
A4	Skruv	Röd	14	Stockholm
A5	Kam	Blå	12	Köpenhamn
A6	Kugge	Röd	19	Stockholm

LevArt

LevNr	ArtNr	Kvantitet
L1	A1	300
L1	A2	200
L1	A3	400
L1	A4	200
L1	A5	100
L1	A6	100
L2	A1	300
L2	A2	400
L3	A2	200
L4	A2	200
L4	A4	300
L4	A5	400



LevNr	Status
L3	30
L2	10

Sökning i flera tabeller

Åtkomst från mer än en tabell samtidigt.

Fråga: För varje artikel, ta fram artikelnummer och städer för motsvarande leverantörer som levererar dessa artiklar

```
SELECT  UNIQUE  ArtNr,Stad
FROM            LevArt,Leverantör
WHERE           LevArt.LevNr=Leverantör.LevNr
```

Resultat:

Leverantör

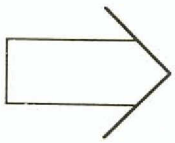
LevNr	LevNamn	Status	Stad
L1	Svensson	20	Stockholm
L2	Nilsen	10	Köpenhamn
L3	Hansen	30	Köpenhamn
L4	Olsson	20	Stockholm
L5	Mäki	30	Helsinki

Artikel

ArtNr	Artnamn	Färg	Vikt	Stad
A1	Mutter	Röd	12	Stockholm
A2	Bult	Grön	17	Köpenhamn
A3	Skruv	Blå	17	Oslo
A4	Skruv	Röd	14	Stockholm
A5	Kam	Blå	12	Köpenhamn
A6	Kugge	Röd	19	Stockholm

LevArt

LevNr	ArtNr	Kvantitet
L1	A1	300
L1	A2	200
L1	A3	400
L1	A4	200
L1	A5	100
L1	A6	100
L2	A1	300
L2	A2	400
L3	A2	200
L4	A2	200
L4	A4	300
L4	A5	400



ArtNr	Stad
A1	Stockholm
A1	Köpenhamn
A2	Stockholm
A2	Köpenhamn
A3	Stockholm
A4	Stockholm
A5	Stockholm
A6	Stockholm

Sökningar med nästlade villkorliga sökningar

Fråga: Leverantörsnamn för leverantörer som levererar artikel med nummer A2

```
SELECT  LevNamn
FROM    Leverantör
WHERE   LevNr       IN
        (SELECT  LevNr
        FROM     LevArt
        WHERE    ArtNr='A2')
```

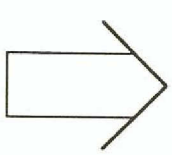
Resultat:

Leverantör

LevNr	LevNamn	Status	Stad
L1	Svensson	20	Stockholm
L2	Nilsen	10	Köpenhamn
L3	Hansen	30	Köpenhamn
L4	Olsson	20	Stockholm
L5	Mäkl	30	Helsinki

Artikel

ArtNr	Artnamn	Färg	Vikt	Stad
A1	Mutter	Röd	12	Stockholm
A2	Bult	Grön	17	Köpenhamn
A3	Skruv	Blå	17	Oslo
A4	Skruv	Röd	14	Stockholm
A5	Kam	Blå	12	Köpenhamn
A6	Kugge	Röd	19	Stockholm



LevNamn
Svensson
Nilsen
Hansen
Olsson

LevArt

LevNr	ArtNr	Kvantitet
L1	A1	300
L1	A2	200
L1	A3	400
L1	A4	200
L1	A5	100
L1	A6	100
L2	A1	300
L2	A2	400
L3	A2	200
L4	A2	200
L4	A4	300
L4	A5	400

Sökning med nästlade nivåer

Fråga: Ta fram namn på leverantör som levererar minst en röd artikel

```
SELECT  LevNamn
FROM    Leverantör
WHERE   LevNr       IN
        (SELECT  LevNr
        FROM     LevArt
        WHERE    ArtNr       IN
                 (SELECT  ArtNr
                 FROM     Artikel
                 WHERE    Färg='Röd'))
```

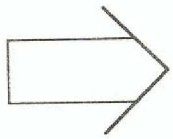
Resultat:

Leverantör

LevNr	LevNamn	Status	Stad
L1	Svensson	20	Stockholm
L2	Nilsen	10	Köpenhamn
L3	Hansen	30	Köpenhamn
L4	Olsson	20	Stockholm
L5	Mäki	30	Helsinki

Artikel

ArtNr	Artnamn	Färg	Vikt	Stad
A1	Mutter	Röd	12	Stockholm
A2	Bult	Grön	17	Köpenhamn
A3	Skruv	Blå	17	Oslo
A4	Skruv	Röd	14	Stockholm
A5	Kam	Blå	12	Köpenhamn
A6	Kugge	Röd	19	Stockholm



LevNamn
Svensson
Nilsen
Olsson

LevArt

LevNr	ArtNr	Kvantitet
L1	A1	300
L1	A2	200
L1	A3	400
L1	A4	200
L1	A5	100
L1	A6	100
L2	A1	300
L2	A2	400
L3	A2	200
L4	A2	200
L4	A4	300
L4	A5	400

Som synes erbjuder SQL en möjlighet att ställa omfattande och komplexa frågor. Dock torde framgå att ju mer komplicerade frågor som ställs desto svårare blir det att uttrycka detta i SQL på ett syntaktiskt korrekt sätt. En fråga som i naturligt språk är lätt att formulera kräver i SQL både god kännedom om SQL:s syntax och fullständig kunskap om det bakomliggande logiska schemat. Det torde framgå att en användare med mycket ytliga kunskaper i SQL knappast kan generera några frågor av högre komplexitet och erhålla förnuftiga svar. Antingen sker fel redan vid syntaxkontrollen eller finns en överhängande risk att de svar som presenteras inte besvarar den fråga som användaren trodde sig ställa. Detta är något som också har visat sig gälla i praktiken.[5]

Hjälp till att fråga

Det finns idag ett antal språk för att manipulera databaser varav SQL här har beskrivits. Problemet med SQL och liknande språk är, som tidigare nämnts, de stora krav på exakthet och korrekthet som ställs hos frågeställaren samt förutsättningen om att en djup kunskap om det logiska schemat föreligger. För att även låta användare som inte uppfyller dessa krav kommunicera med ett IS behövs andra typer av gränssnitt eller ytterligare funktionalitet hos existerande gränssnitt. Iden är således att bygga in hjälpmekanismer i ett frågespråk, i detta fall SQL, som gör det möjligt för sporadiska IS-användare att använda systemet. Hur detta hjälpsystem kan se ut och vilka funktioner som bör ingå beskrivs i avsnitt 4 och 5.

---

[5] (Re81)

3. Användaranvändning

I detta avsnitt ges en kort översikt av några befintliga typer av gränssnitt samt deras utmärkande drag.

Med gränssnitt, (eng interface), menas den del av ett datorsystem som en användare nyttjar för att kommunicera med systemet. För att en kommunikation skall vara meningsfull måste en fråga som ställs till systemet omedelbart ge någon slags respons, antingen i form av ett direkt svar på frågan eller ett meddelande om att svaret dröjer ett tag och sedan presentera svaret. Detta ställer således krav på att viss typ av utrustning måste ingå i datorsystemet.[6] I dagsläget används huvudsakligen bildskärmar som media för ett systems gränssnitt även om andra möjligheter finns t ex via mikrofon och högtalare. Tidigare skedde "kommunikationen" via hålkorts- eller remsläsare och pappersutskrifter. Dock är det i hög grad tveksamt om man kan tala om en kommunikation i den bemärkelse som här avses.

Gränssnittet kan vara utformat på olika sätt och nedan ges en kort beskrivning av en del olika principer för utformning av gränssnitt som förekommer idag. Notera att det är grundprinciper som beskrivs. Många existerande gränssnitt lånar ideer från andra typer och blir ett slags hybrider mellan olika varianter.

De olika typer av gränssnitt som kan ingå i ett hjälpsystem för SQL presenteras närmare i avsnitt 5.

Kommandostyrning

Två typer av kommandobaserade gränssnitt kan urskiljas, formella språk och naturligt språk. Vad som skiljer dessa åt är den grad av formalism som krävs av en användare. I formella språk måste en fråga vara fullständigt korrekt ställd och ofta finns endast ett sätt att ställa en viss fråga. I naturligt språk är kravet på exakt formulering inte lika strängt dvs samma fråga kan uttryckas på olika sätt. Naturligt språk kommer att behandlas i avsnitt 5.

I ett gränssnitt, där formella språk används, är det användaren av systemet som är den aktiva parten och som i varje ögonblick talar om för systemet vad som skall göras. För att kunna styra systemet på ett korrekt sätt ställs stora krav på att användaren känner till vilka kommandon som existerar samt vilka konsekvenser ett givet kommando för med sig. Ett flertal sk operativsystem är uppbyggda enligt denna princip. SQL kan med fog placeras in i denna grupp av gränssnitt.

Ett exempel på kommandostyrd dialog kan vara att vi vill ta bort en person ur ett register. I SQL kan detta göras med ett DELETE-kommando.

```
Kommando> DELETE leverantör WHERE LevNr=L3
```

Om allt går väl raderas uppgifter om denna person och systemet anger att det väntar på nästa kommando genom att skriva:

```
Kommando>
```

---

[6] (Na77)

Menyer

I ett menystyrt gränssnitt visas i alla lägen en meny dvs en lista med aktuella valmöjligheter för en användare. Genom att välja ett visst alternativ av de i menyn presenterade valmöjligheterna kommer antingen en ny meny att visas eller ett kommando att utföras. Denna typ av gränssnitt har visat sig fungera utmärkt för användare utan datorvana eller med ringa kännedom om det aktuella system som de för tillfället använder. Mer erfarna användare upplever efter en tids användning när de behärskar systemet till fullo det frustrerande att mer eller mindre tvingas gå igenom en kedja av menyer när de istället direkt hade kunnat ge ett kommando.

Ofta förekommer sk hjälpmenyer, dvs menyer som kan visas för en användare närhelst denne så önskar och som innehåller hjälp- eller informationstexter.

Vi använder ovanstående exempel för att belysa principen. Som regel presenteras en sk huvudmeny först:

```
Huvudmeny

    Skapa 1
    Förändra 2
    Radera 3
    Lista 4
    Sluta S

    Välj alt.....3
```

Fig 3.1 Menyexempel

Som åtgärd anges 3 varvid nästa meny presenteras:

```
Radera

    Leverantör 1
    Artikel 2
    LevArt 3
    Sluta S

    Välj alt.....1
```

Fig 3.2 Menyexempel

Denna gång anges 1 och nästa meny presenteras:

```
Radera Leverantör

Ange vilken leverantör som skall raderas: L3
```

Varvid den sist presenterade menyn återigen visas.

Grafik

Grafiska gränssnitt kräver att det finns sk grafiska bildskärmar till skillnad från de ovan nämnda typerna som nöjer sig med vanliga textskärmar. Skillnaden är hur hög upplösning eller hur många punkter som skärmen består av. Genom att använda grafiska skärmar kan bilder eller olika symboler presenteras. Till grafiska skärmar kopplas ofta en sk mus, en liten ask som är kopplad till skärmen och genom att flytta musen flyttas samtidigt en symbol på skärmen. När symbolen som representerar musen befinner sig på lämplig plats på skärmen, det kan vara på en figur av något slag, och användaren trycker på en knapp på musen, kommer ett kommando att utföras. Principen är således att istället för att tex skriva ordet "leverantör" på skärmen ritas en figur som föreställer en leverantör. Denna typ av gränssnitt kallas ikoner. Fördelen med dessa system är framför allt att man slipper använda tangentbordet i särskild stor omfattning. Vidare finns en inneboende slagkraft i bilder som det skrivna ordet i vissa lägen har svårt att konkurrera med.

Notera att ovanstående endast avsåg grafisk utmatning. Grafisk inmatning, dvs system där en användare själv ritar en symbol eller figur och systemet förmår analysera figuren på ett meningsfullt sätt, befinner sig fortfarande på forskningsstadiet. Däremot är det fullt möjligt att rita bilder bestående av olika figurer på ungefär samma sätt som med papper och penna och använda dator som bildredigerare. Men notera att datorsystemet inte gör någon analys av bilden som sådan. Grafiska gränssnitt i samband med HSQL beskrivs närmare i avsnitt 5.

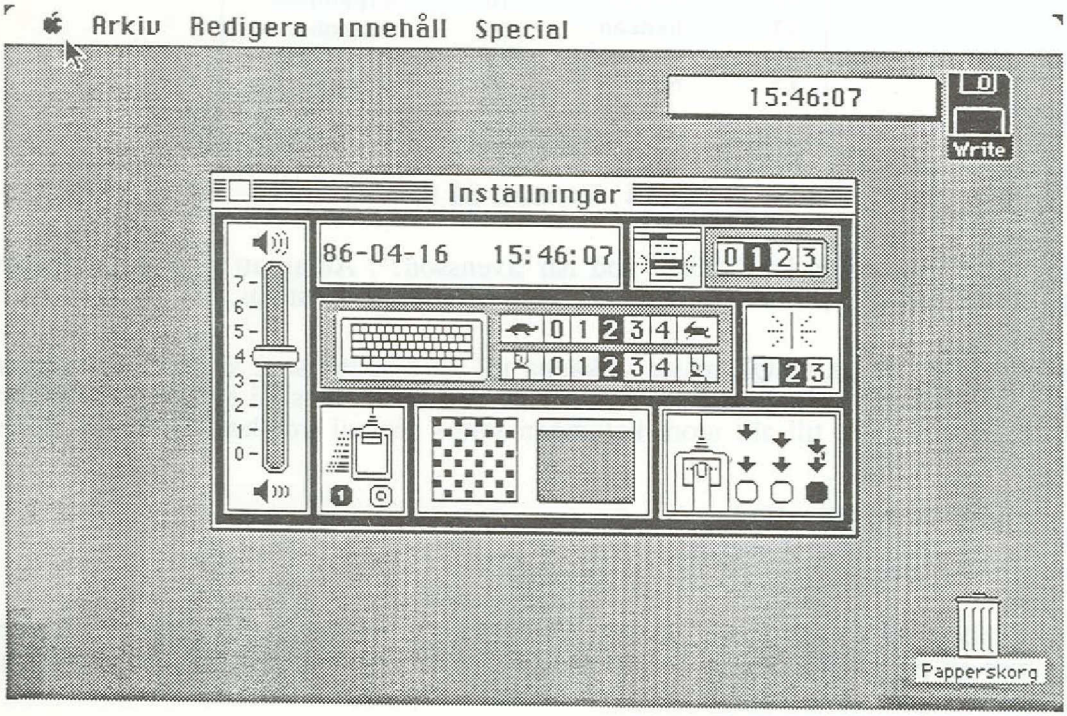


Fig 3.3 Ett exempel på en grafisk skärmutskrift

Andra gränssnitt

En variant av grafiska gränssnitt är att i stället för att använda en mus pekar man med ett finger, ljuspenna eller dylikt direkt på skärmen som därmed aktiveras på något sätt.[7]

Redan idag finns system där man i stället för att ange kommandon till systemet via ett tangentbord talar in kommandot via mikrofon. På motsvarande sätt kan systemet svara med syntetiserat tal via högtalare.

Användarvänlighet?

Gemensamt för de ovan nämnda typerna av gränssnitt är att om en användare vill utföra ett kommando som inte kan tolkas korrekt, men som ändå i någon mening är förnuftigt ur användarens synpunkt, resulterar det oftast i ett felmeddelande.

Ett exempel kan vara på sin plats. Samma databasexempel som tidigare antas.

Leverantör

LevNr	LevNamn	Status	Stad
L1	Svensson	20	Stockholm
L2	Nilsen	10	Köpenhamn
L3	Hansen	30	Köpenhamn
L4	Olsson	20	Stockholm
L5	Mäki	30	Helsinki

Fig 3.4 Exempel på tabell

Om frågan som ställs är: "Vilken kod har Svensson?". Antag att vi som användare inte känner till att kolumnen som koden finns lagrad i heter status.

Om vi ställer frågan i SQL måste vi känna till att tabellen som avses är leverantör, vidare att kod inte är kod utan status samt att Svensson finns i LevNamn-kolumnen. Antag att vi känner till allt utom kolumnen status som vi tror heter kod och ställer frågan:

```
SELECT   Kod
FROM     Leverantör
WHERE    LevNamn='Svensson'.
```

Vad händer? Helt klart är att 20 inte kommer att presenteras som resultat. Snarare kommer ett felmeddelande att genereras om att kod inte existerar som kolumnnamn. Samma sak gäller om fel tabell eller om tabeller som inte existerar används.

Vidare finns problemet med kryptiska kolumnnamn. Är det tex självklart att ArtNr står för artikelnummer?

---

[7] (Se77)

Dessa problem kan till en viss del undvikas i menysystem men andra problem kan uppstå. I ett menysystem kanske inte möjligheten att söka ut leverantörers status via deras namn förekommer som alternativ i någon meny.

Vad man vill åstadkomma hos ett datorsystem är att oberoende av gränssnittet skall det under ytan finnas en mekanism som gör det möjligt att försöka tolka en användares kommando eller fråga på ett semantisk förnuftigt sätt. Med andra ord, en kommunikation mellan ett system och en användare skall i princip fungera på samma sätt som om den utspann sig mellan två människor! Och det är här som kunskapsteknik kommer in i bilden.[8]

4. KUNSKAPSBASERADE SYSTEM

Kapitlet ger en allmän beskrivning av vissa centrala begrepp inom kunskapsteknik och AI (Artificiell intelligens) samt relaterar detta till HSQL. Vidare specificeras en tänkbar konfiguration och funktionalitet hos HSQL.

Kunskapsteknik

Försöken att bygga in intelligens i datorer har pågått med varierande intensitet i drygt trettio år. De senaste tio-femton åren har både forskningsinsatsen och forskningsresultaten varit betydande inom området AI. Nya tekniker och metoder har utvecklats vilket möjliggjort byggandet av datorsystem som uppvisar ett intelligent beteende.

Det finns i denna artikel inte utrymme för att detaljredovisa de tekniker som utvecklats och de områden där dessa tekniker används. Det finns en ansenlig mängd litteratur på området varav kan nämnas[9] som ger en lämplig introduktion till ämnet.

Helt kort kan nämnas några områden där AI-teknik använts:

- naturligt språk förståelse
- intelligent informationsåtervinning
- expertsystem
- robotics

Några av de tekniker som utvecklats är:[10]

- olika sökmetoder
- kunskapsrepresentation
- heuristiskt resonemang
- programspråk och programmeringsverktyg

---

[8] (Pi83).
[9] (Ni80)
[10] (Or83)

Med ett kunskapsbaserat system menas ett system där kunskap finns representerad explicit i någon form samt mekanismer för att resonera kring och om denna kunskap.

Allmänt kan sägas att traditionell databehandling utgörs av programmerade algoritmer som hanterar data medan AI-program snarare resonerar om och förändrar en kunskapsbas [11] samt snarare symboliskt manipulerar kunskap än hanterar data.

Expertsystem

En variant av kunskapsbaserade system är expertsystem. Med expertsystem avses skräddarsydda dataprogram som beter sig som mänskliga experter inom ett begränsat problemområde.[12]

Ett expertsystems beteende skall efterlikna en mänsklig experts, vilket rent konkret innebär att bl a följande komponenter skall ingå i systemet:[13]

- kunskapsbas
- konsultationssystem
- förklaringsmekanismer
- inferensmaskin

Expertsystem har en aktuell domänkunskap som motsvarar den mänsklige expertens kunskap eller en samlad expertis kunskap inom området samt mekanismer för att resonera kring denna kunskap samt rutiner för att kommunicera med en användare av systemet. En användare av ett expertsystem kan vara en expert som vill testa eller kontrollera sin lösning av ett problem eller en icke-expert som vill få ett problem löst på ett lika adekvat sätt som om det vore en mänsklig expert som stod för problemlösningen.

Hjälpsystem

Det hjälpsystem som skall ingå i HSQL kan beroende på vald ambitionsnivå och funktionalitet implementeras på olika sätt. Dock kan sägas redan nu att de centrala delarna lämpligast realiseras som ett kunskapsbaserat system genom att välja lämpliga delar av befintlig kunskapsteknik. Då vissa egenskaper som normalt ingår i expertsystem inte tas med i HSQL bör hjälpsystemet snarare benämnas som ett kunskapsbaserat hjälpsystem än ett expertsystem.

Beroende på ambitionsnivå och funktionalitetsgrad hos HSQL kommer systemet att erhålla olika utseende. Det innebär att några av de nedan presenterade funktionerna helt eller delvis kan komma att utgå. Vilka konsekvenser ett visst gränssnitt för med sig anges i avsnitt 5. Det är dock vår avsikt att ange en, som vi anser, lämplig hög ambitionsnivå och föreslå en arkitektur för detta. Denna ambitionsnivå skall så långt som det är möjligt vara oberoende av senare vald ansats även om det inte är möjligt

[11] (So84)
[12] (Wa85)
[13] (Fe81)

att totalt bortse från aktuell ansats vid en slutlig implementering.

Det ingår ett antal komponenter i HSQL och dessa beskrivs i detalj nedan.

Tidigare har nämnts att en viktig beståndsdel av ett kunskapsbaserat system är representationen av kunskap. Det är således av vikt att välja en sund och enhetlig representation för den kunskap som finns i systemet. För att kunna hantera en fråga som ställs av en användare till systemet måste frågan först transformeras om från den representation som den har i gränssnittet till den interna representationen.

En mänsklig analogi kan vara att en person med svenska som modersmål visserligen kan ta emot en fråga på främmande språk, tex engelska, men som regel resonerar personen mentalt på sitt eget modersmål, dvs frågan översätts först till svenska innan en analys av frågan äger rum. För HSQL innebär det att oberoende av gränssnitt måste denna transformering alltid ske. Denna transformeringsprocess är en del av en mer omfattande process som benämns tolkning, (eng parsing).

Med parsing avses således både analys av syntax och semantik. Notera att vi i denna framställning presenterar de olika momenten i HSQL som om de utförs i separata funktioner och i sekvens. Detta är en förenklad bild eftersom det snarare är så att vissa moment utförs i en och samma fas än sekventiellt. Vi har dock valt att presentera de logiskt urskiljbara funktionerna som separata enheter av pedagogiska skäl. I exemplen används huvudsakligen frågor i naturligt språk. Detta är också en medveten förenkling. Dock är det bakomliggande resonemanget giltigt för andra typer av gränssnitt.

Syntaxanalys

Den tolkning som sker först är en syntaktisk tolkning, dvs en kontroll av att frågan är syntaktiskt korrekt ställd äger rum. Detta innefattar kontroll av stavning, ordföljd etc. Ambitionsnivån skall vara att inte generera felmeddelanden vid syntaktiskt fel utan dessa skall så långt som möjligt accepteras och rättas av syntaxkontrollen.

Ett exempel kan vara frågan:

Vilka leverantörer finns i Stockholm ohc har status större än 20?

Här är *och* felstavat *(ohc)* vilket inte skall resultera i ett felmeddelande utan systemet skall förstå att det är *och* som avses. En syntaxanalys kan visserligen enbart kontrollera stavning, ordföljd hos de enstaka ord som ingår i språket eller de datatobjekt som ingår i databasen, men bara en sådan relativt enkel mekanism skulle uppskattas av en användare.

Ett annat exempel kan vara att vi använder SQL med svenska ord i stället för engelska. Mönstret för att ställa en SQL-fråga är SELECT... FROM...WHERE... Detta kan på svenska uttryckas som VÄLJ...FRÅN...DÄR...

Om en fråga ställs:

SVÄLJ LevNr FRÅN Leverantör DÄR stad=Oslo

skall denna ändå accepteras trots stavfel av VÄLJ.

För att kunna utföra en syntaxanalys måste en grammatik finnas definierad, dvs en regelbeskrivning över de syntaktiskt korrekta meningar eller frågor som kan ingå i en viss ansats.[14] [15] [16] Om stavfel o dyl skall tillåtas måste dessutom mekanismer finnas som tar hand om felaktiga meningar. Här kan sägas att grammatiken för SQL är tämligen enkel medan den för naturligt språk är mer komplex.

Notera att syntaxanalysen inte enbart analyserar frågan utan även transformerar om frågan till den interna representationen. Om syntaxanalysen ändå misslyckas skall en analys av vilken typ av fel som föreligger ske och användaren skall ges en tydlig och utförlig beskrivning av felsymptomen. Allt för att ge användaren en rimlig chans att ställa en korrekt fråga i nästa försök. Ett system som enbart svarar "Syntaxfel! Försök igen" eller något liknande uppfyller knappast det kravet. Gränsdragning för vad som skall betraktas som stavfel och ordfel är inte helt klar. Principen kan vara att välja en viss nivå och utvärdera efter en viss tids användning och beroende på resultatet av denna flytta gränsen i någon riktning.

Första fasen i hjälpsystemet består i att analysera syntaxen hos frågan samt att transformera den ursprungliga frågan till en intern representation. Frågan som nu är representerad på annat sätt går därefter till en semantisk analys.

Naturlig frågeordning

En nyttig mekanism som bör ingå i systemet hanterar frågor som refererar till icke explicit angivna dataobjekt eller syftningar, s k anaforer och ellipser (tillbakasyftning resp utelämnande).[17] [18] [19] En användare som ställer en följdfråga till en tidigare ställd fråga måste kunna göra detta utan att repetera i stort sett hela föregående fråga.

Exempelvis skall en dialog kunna utspinna sig på följande sätt:

> Hur många leverantörer levererar röda muttrar?
>
> 2
>
> Vilka är dessa?
>
> Svensson
>
> Nilsen

Ofullständiga frågor som syftar tillbaka på tidigare frågor skall tas om hand i syntaxanalysen eller i samma skede. Ett system som klarar frågor som är kopplade till tidigare ställda frågor har en oerhörd genomslagskraft då beteendet blir snarlikt det som människor sinsemellan visar upp vid kommunikation. Normalt utspelar sig ett samtal mellan två individer i ett visst sammanhang. Det är inte så att man byter samtalsämne från en sekund till en annan utan samtalet rör sig inom en viss kontext eller område. Genom att uttnyttja detta faktum kan en kontext skapas.[20] Ett system

---

[14] (TV72)
[15] (PW80)
[16] (Wa84)
[17] (LO71)
[18] (Lu85)
[19] (Wa84)
[20] (Ro84)

av HSQL:s typ bör ha den funktionaliteten att kunna skapa en sådan kontext som sedan kan användas för analys av frågor som är ofullständiga.

Semantisk analys

Den semantiska analysen består i att göra en förnuftsmässig tolkning av en fråga samt att eventuellt göra en ytterligare transformering av den interna representationen av den ursprungliga frågan. Med förnuftsmässig tolkning avses inte bara en kontroll av att frågan är möjlig att besvara utan även analysera varför en fråga eventuellt inte går att besvara.[21] [22] En typ av semantisk analys är kontroll av att frågan behandlar rätt typ av dataobjekt.

Ett exempel. På frågan

Var bor Svensson?

kan följande ske. I det logiska schemat finns uppgifter om leverantörer. Vidare finns en allmängiltig regel eller utsaga i en regelsamling som anger att enbart leverantörer har postadresser.

Om något dataobjekt har en postadress så är det objektet en leverantör.

Det finns också angivet ett antal synonymangivelser där en sådan är att bo någonstans är det samma som att ha en postadress samt att postadress är det samma som stad.

Att bo är synonymt med att ha postadress

Postadress är det samma som stad

Med vetskap om dessa regler och påståenden förmår systemet med hjälp av inferensmaskinen att dra slutsatsen att Svensson måste vara en leverantör och att det således är en utsökning i leverantörstabellen med Svensson som söknyckel i namnkolumnen som skall utföras och att sökt kolumn är stad.

Ett annat exempel där frågan är en aning korthuggen, vem säljer artiklar med nummer A1.

Vem säljer A1?

En titt i kunskapsbasen ger vid handen att endast leverantörer levererar artiklar samt att sälja är synonymt med leverera. Därför måste det vara leverantörer som söks. Eftersom det inte är säkert om det finns en eller flera leverantörer som levererar artikel A1 kan frågan formuleras om till vilka leverantörer levererar artiklar med nummer A1.

Vilka leverantörer levererar artikel med nummer A1?

Som frågan ställdes går den inte att besvara, däremot existerar en förnuftig omtolkning av frågan som gör att vettiga svar kan presenteras. Systemet har således ett val

[21] (Am85)

[22] (Wa84)

att göra. Skall frågan förkastas eller skall systemet presentera den tolkning som systemet har gjort och fråga användaren om utsökning fortfarande skall göras? En rimlig ambitionsnivå är att systemet talar om vilken tolkning som gäller och ger användaren chans att godkänna eller förkasta förslaget.[23] [24]

En semantisk analys innebär således att det logiska schemat måste användas för tolkning, vidare måste det finnas en uppsättning generella regler som gäller den aktuella databasen, tex att alla bilar har en ägare. Dessutom måste det finnas ett synonymlexikon med de vanligaste synonymerna till kolumnnamn och tabellnamn, tex att stad är samma sak som postadress. Dessa komponenter ingår i det som kallas kunskapsbasen. En annan benämning som omfattar samtliga komponenter är konceptuellt schema.[25]

Ett allvarligt problem med semantisk analys är att det ofta går att tolka en mening på flera olika sätt, sk ambiguitet. För att lösa detta kan i huvudsak två principer användas. Den första varianten presenterar samtliga tolkningar för användaren som får välja vilken som avses. Den andra varianten försöker gissa sig till vilken tolkning som verkar rimlig. För att klara av detta kan en uppsättning s k heuristiska regler definieras som hjälper systemet att välja rätt tolkning.[26] Det kan närmast liknas vid tumregler som kan basera sig på statistik eller erfarenhet om vilka frågor som oftast ställs av användare.

En högre ambitionsnivå kan vara att dessutom tillhandahålla en uppsättning regler som närmast skall betraktas som "sunt förnuft". Det är regler som inte är specifika för databasens innehåll utan allmängiltiga regler som existerar i vardagslivet.

I princip kan all kunskap formaliseras, frågan är var gränsen skall dras. Vad som i detta fall menas med "sunt förnuft" är regler och utsagor som kan placeras i denna gränszon.

Databasanrop

Resultatet av den semantiska analysen går vidare till ett system som genererar SQL-satser från den interna representationen. Detta system bygger upp SQL-anrop och anropar ett befintligt databashanteringssystem. De SQL-anrop som sker i detta ögonblick skall aldrig kunna generera ett felaktigt anrop pga syntaxfel eller orimlig semantik. Detta är ju en grundförutsättning för ett intelligent hjälpsystems funktionalitet.

Notera att det är ett befintligt relationsdatabashanteringssystem som skall anropas av hjälpsystemet. Det kan således vara vilket DBHS som helst som tillåter SQL-anrop. Vidare är det möjligt att generera frågor till vilken typ av frågespråk som helst. Anledningen till att begränsa diskussionen till SQL är som tidigare nämnts dess relativt stora spridning samt det standardiseringsarbete som pågår.

---

[23] (BMS84)
[24] (Am85)
[25] (BL84)
[26] (Ni80)

Svarsanalys

Resultatet av DBHS-anropet kan sedan hanteras på olika sätt beroende på vilket svar som genereras. Om ett positivt svar blir resultatet, dvs en eller flera relationselement utgör svaret på frågan, presenteras detta för frågeställaren samtidigt som kontextrepresentationen förändras.

Om ett negativt svar erhålls, dvs några relationselement som uppfyller frågan har inte funnits, bör en analys av frågan ske och om möjligt formulera om frågan och ånyo anropa DBHS.[27] [28] [29]

Ett exempel på denna metod kan vara:

Finns det några blåa skruvar?

Om det nu inte finns några blåa skruvar blir svaret nej. Vad som däremot borde vara av intresse för frågeställaren att få vetskap om är att det finns röda skruvar. För en inköpare kan denna uppgift vara att föredra framför ett nej-svar.

Denna typ av beteende hos ett system brukar benämnas samarbetsvilja, (eng cooperative response).[30] Denna mekanism är i hög grad fortfarande föremål för forskningsverksamhet men bör så långt som möjligt finnas inbyggt i ett framtida HSQL.

Denna analys sker i svarsmottagaren från DBHS-anropet och skall formulera om frågan och vidarebefordra denna till funktionen som genererar DBHS-anrop. Detta sker enbart om negativa svar erhålls och en omtolkning med hjälp av tex frågegeneralisering är möjlig.

Sammanfattning av kunskapsbaserade system

I det här avsnittet har presenterats en möjlig konfigurering av HSQL utifrån ett språk och gränssnittoberoende perspektiv och ambitionsnivå vad gäller funktionalitet.

Vissa av de funktioner som presenterats kan dock svårligen implementeras till alla ansatser men de flesta måste med nödvändighet ingå. En mer detaljerad beskrivning av de olika ansatserna sker i det följande avsnittet.

---

[27] (Hi84)
[28] (Ma84)
[29] (Mo84)
[30] (TC85)

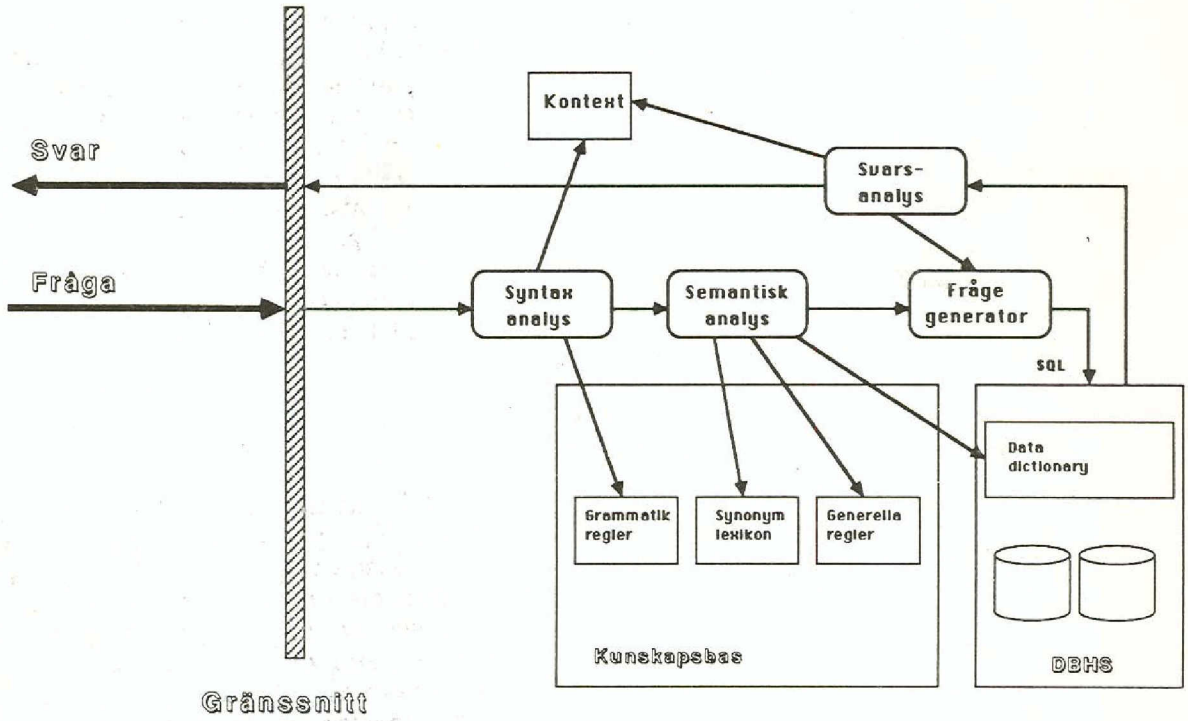


Fig. 4.1 Schematisk bild av HSQL

5. Olika hjälpsystem för SQL

Tidigare har beskrivits en tänkbar konfiguration och funktionalitet hos ett kunskapsbaserat hjälpsystem för HSQL. Diskussionen fördes utifrån ett gränssnittsoberoende perspektiv. I detta avsnitt beskrivs fem olika typer av ansatser som samtliga utgör tänkbara kandidater som gränssnitt i HSQL. Tanken har varit att beskriva grundprincipen för de olika gränssnitten samt beskriva något representativt befintligt system, om uppgifter om något sådant funnits. Vidare anges skillnader i funktionalitetskrav som ställdes på HSQL i föregående avsnitt. I texten görs kommentarer angående krav på kringutrustning. Vi har valt att utgå från något som kallas basutrustning, dvs det finns en textskärm med tillhörande tangentbord med programmerbara funktionstangenter anslutna till någon datorenhet. I den mån någon ansats kräver utrustning utöver detta anges detta explicit.

SQL

Att använda SQL självt som gränssnitt i HSQL innebär att de mekanismer, primitiver och dataåtkomstmöjligheter som finns i SQL bibehålls. Dock kommer systemet att utökas med hela funktionaliteten hos ett kunskapsbaserat system som det tidigare har beskrivits. SQL:s syntax skall bibehållas även om det givetvis inte behöver vara engelska ord som ingår. Varje nordiskt land kan ha sitt eget språk som aktuellt gränssnitt. Själva grundstrukturen för en fråga i SQL kommer att kvarstå. I SQL är formen SELECT...FROM...WHERE. Detta mönster bibehålls men kan uttryckas i ett annat lämpligt språk, VÄLJ...FRÅN...DÄR. Samtliga primitiv för t ex sortering, genomsnittsberäkning etc kommer att bibehållas men även här kan språket vara ett annat. Notera att det således inte enbart är fråga om att byta ut de engelska orden i SQL mot tex finska ord. Det innebär att gränssnittet dessutom utökas med de funktioner som angetts i föregående avsnitt.

Fördelar med SQL

Fördelar med att bibehålla SQL som gränssnitt är att man utnyttjar de funktioner som ingår i det kunskapsbaserade hjälpsystemet. För syntaxens del innebär det att systemet tillåter ett stort inslag av uppenbara felstavningar samt delvis felaktig ordföljd. Syntaktiskt felaktiga SQL-frågor formuleras om av systemet, om det är möjligt, och presenteras för användaren som ges en möjligt att acceptera eller förkasta omformuleringen Det har visat sig att en stor del av de fel som görs av SQL-användare är av syntaktisk karaktär och det vore av stort värde och hjälp för en användare om systemet vore mer benäget att acceptera smärre syntaktiska felaktigheter.

Semantiska analysen bör kunna utnyttjas till fullo liksom svarsanalysen. Till en viss del bör även gränssnitten kunna bli kontextkänsliga, dvs en ställd fråga behöver inte vara fullständig utan syftning till tidigare ställd fråga skall kunna göras.

Om en fråga har ställts:

```
VALG    Status
FRA     Leverantör
DAA     Navn='Nils Nilsen'
```

bör nästa fråga vara möjlig att ställa:

```
VALG    By OG LevNr
```

och systemet skall inse att det är leverantörstabellen som avses med rader där namn = Nils Nilsen.

Huruvida det är möjligt att till fullo utnyttja de mekanismer som ingår i ett kontextkänsligt system är tveksamt. Här krävs en djupare undersökning än vad som här är möjligt.

Ambiguitet skall hanteras på likartat sätt. Om det vid analysen visar sig att flera tolkningar är möjliga skall dessa presenteras för användare som ges möjlighet att välja någon tolkning eller förkasta samtliga.

Nackdelar med SQL

Den allvarligaste nackdelen som kan identifieras är att även om både syntaktiska och semantiska analyser gör systemet mer användarvänligt är det tveksamt om det hjälper de användare som idag har störst problem med SQL. Som tidigare har nämnts krävs en relativt djup kunskap om databasens innehåll eller det aktuella logiska schemat för att kunna ställa förnuftiga frågor i SQL. De användare som är i behov av mest hjälp är just de som saknar denna insikt om databasens innehåll eller har en oklar bild av densamma. Detta förhållande kan mildras bl a med hjälp av den semantiska analysen och synonymlexikonet. Dock kvarstår att SQL:s syntax är artificiell och denna kan man inte helt komma förbi trots samarbetsvilja hos systemet.

Krav

Eftersom SQL inte ställer några särskilda krav på kringutrustning, bör den typ av gränssnitt som här skissats kunna realiseras i alla system som i dag har SQL.

Införande av SQL-baserat system

Några idag existerande system av den typ med SQL som gränssnitt som ovan beskrivits finns ej vad vi vet. Arbetsinsatsen för att implementera detta gränssnitt underlättas avsevärt då syntaxen är relativt enkel samtidigt som det inte är nödvändigt att till fullo realisera samtliga funktioner för att få något användbart. Om avsikten är att införa samtliga funktioner är arbetsinsatsen för dessa i princip densamma för samtliga ansatstyper.

Sammanfattning - SQL

Ovan har beskrivits SQL som tänkbart gränssnitt i HSQL. Beroende på perspektiv kan olika slutsatser dras. Om syftet med HSQL är att ytterligare underlätta för den grupp som idag använder SQL, samt i viss mån utöka den användarkategorin, är denna typ av ansats lämplig. Om syftet är att i stort sett alla skall kunna använda HSQL, dsv även de som idag inte klarar av SQL, är det i hög grad tveksamt om denna gränssnittstyp är lämplig.

Grafik

Tyvärr finns ingen klart uttalad norm om vad som menas med ett grafiskt gränssnitt i detta sammanhang.[31] [32] I viss mån blir denna framställning stipulativ såtillvida att vi utgår från några olika ideer som presenterats huvudsakligen i forskningsartikeler och sammanställer dessa till en idealiserad bild av hur vi anser att ett grafiskt gränssnitt kan se ut.

I princip är en grafisk framställning av en relationsdatabas en bild med symboler för tabeller och kolumner. För att ange vilka symboler som logiskt hör ihop ritas vanligen pilar mellan symbolerna. Man brukar i databassammanhang skilja på intension och extension hos en databas. Intensionen motsvaras i relationsdatabasen av kolumnernas namn, extensionen hos de aktuella värden eller data som finns i kolumnerna. Vanligen presenteras en bild av intensionen hos databasen eller tabellen.[33]

---

[31] (Pi83)

[32] (Sp84)

[33] (BL84)

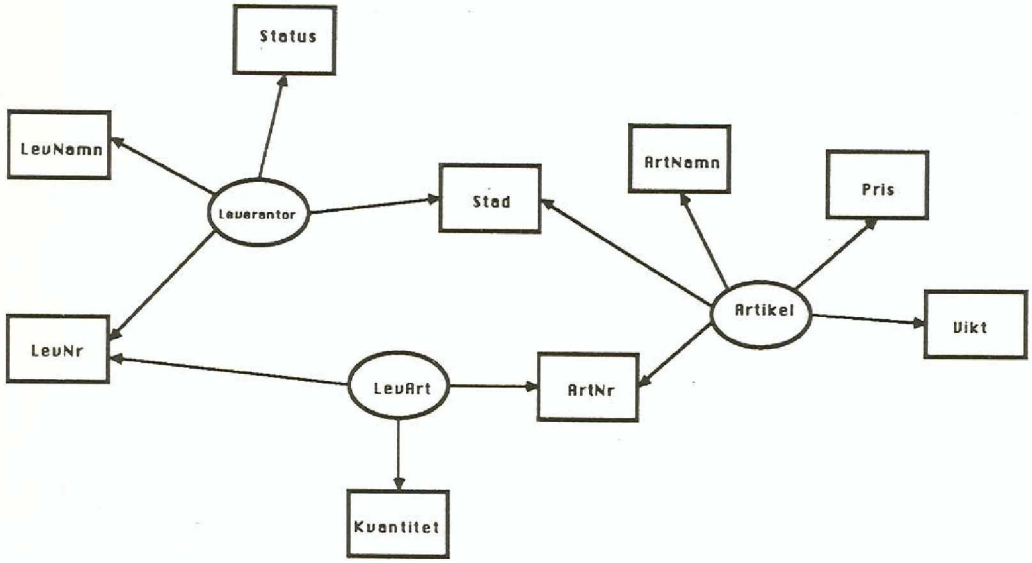


Fig 5.1 Grafisk bild av logiska schemat

I figur 5.1 visas en bild av tre tabeller, leverantör, artiklar och levart. Deras intentioner dvs LevNr, ArtNr etc länkas samman med hjälp av pilar. Vilka symboler som används är egalt; huvudsaken är att de används konsekvent och att semantiken är klar. Till bilden visas som regel ett antal menyer som används för att ange tex restriktioner på ett visst kolumnvärde.

Test	Logik	Mängd
mindre än	och	räkna
mindre eller lika med	eller	summera
lika med	icke	genomsnitt
större än		max
större eller lika med		min

Fig 5.2 Exempel på menyer i ett grafiskt gränssnitt

En användare kan, genom att flytta markören på skärmen till en viss symbol, ge ett kommando som anger att den underliggande tabellens innehåll skall visas. Alternativt kan en markering göras vid en viss symbol och denna kommer då att t ex lysas upp eller börja blinka eller på annat sätt ange att den är aktiverad. Genom att på detta sätt aktivera olika symboler som en användare är intresserad av kan sedan en utsökning ske genom att placera markören på ett lämpligt kommando, t ex visa underliggande tabellinnehåll.

Genom att välja symboler ur olika relationer erhålls motsvarigheten till en sammanslagning av tabeller i SQL. Komplexa villkor kan kopplas till varje symbol som är aktiverad.

Till exempel, om status har aktiverats i figur 5.1 kan ett villkor anges att status skall vara mindre än 30.

Att ställa en fråga består således i att vandra omkring i den grafiska bilden av databasen och aktivera olika symboler samt ange restriktioner till dessa och slutligen meddela systemet om att visa resultatet.

Fördelar med grafik

Ett antal fördelar med en grafisk gränssnitt kan identifieras.[34] [35]

Eftersom namn visas på aktuella kolumner och tabeller föreligger ingen risk för syntaktiska fel från användaren. Användarens behov av att använda tangentbordet reduceras till ett minimum eftersom val sker med mus och kommandon visas i befintliga menyer. I en lokal del av det logiska schemat föreligger naturliga samband mellan symbolerna. Semantiken är förhållandevis klar i varje begränsad del av schemat. En kontext kan lätt realiseras genom att låta alla aktiverade symboler förbli aktiva efter en utsökning tills en explicit deaktivering sker.

Nackdelar med grafik

Vid utsökningar, som omfattar kolumner som är vitt skilda i ett omfattande logiskt schema, tenderar en användare att snabbt förlora orienteringen. Man brukar tala om att navigera i en databas och ju större och mer komplex en databas är desto svårare blir navigeringen. Endast en begränsad del av det logiska schemat kan visas samtidigt på skärmen. Trots att tangentbord inte behöver användas i någon större utsträckning måste inte desto mindre restriktioner anges via tangentbord. Ett alternativ kan vara att utöka bilden med alltfler menyer. Risken är då att bilden som presenteras blir oläslig pga att alltför mycket information samtidigt måste presenteras för en användare. Den ordning som symboler aktiveras i får betydelse för resultatet av en utsökning. Semantiken kan därmed bli oklar.

Krav

Grafiska gränssnitt kräver självfallet att någon form av grafiska skärmar används. Vidare måste antingen en mus eller liknande anslutas eller funktionstangenter med motsvarande funktionalitet programmeras. Av de system som finns idag har ett flertal implementerats på s k arbetsstationer. Dessa kan närmast betraktas som mycket kraftfulla persondatorer med möjlighet till avancerad grafisk bildbehandling.

---

[34] (Se77)

[35] (Ka85)

Införande av grafikbaserade system

Ett antal forskningsprojekt pågår som har till syfte att realisera grafiska gränssnitt mot relationsdatabaser och SQL. Dessa projekt pågår ännu och några kommersiella produkter finns i dagsläget inte att tillgå. Bland de system som är under utveckling eller har utvecklats kan nämnas: LINDA med VQG,[36] se bilaga 1, CUPID,[37] FORAL LP,[38] LID,[39] Sembase med Ski,[40] CONCEPT D/CS.[41]

Ur implementeringssynpunkt kan ett HSQL baserat på ett grafiskt gränssnitt delas in i två etapper, gränssnitt och hjälpsystem. Första etappen består i att realisera själva gränssnittet som säkerligen kan vara användbart utan några hjälpmekanismer. Systemet skulle i detta skede vara helt utan intelligens. Hjälpfunktionerna skulle sedan kunna implementeras i etapp två.

Sammanfattning - grafik

Grafiska gränssnitt med den tolkning vi har gjort är en klart intressant och lovande teknologi som utgör ett lockande alternativ till de övriga ansatserna. I och med att principen inte realiserats i kommersiella system finns dock farhågor att tekniken inte lever upp till den bild som presenterats. Något som ytterligare talar emot är de särskilda krav på extra kringutrustning som med nödvändighet föreligger. Databaser är som regel komplexa och omfattande vilket kommer att avspegla sig i ett stort logisk schema. De navigationsproblem och de semantiska oklarheter som härmed kan uppstå är likaledes ett oklart område.

Naturligt språk

Med ett gränssnitt baserat på naturligt språk avses ett gränssnitt där en användare kommunicerar med ett system på sitt eget modersmål. Detta innebär att det inte behövs några artificiella konstruktioner eller ord i en fråga. Det kan vara finska, danska eller vilket annat existerande språk som helst som används. En användare formulerar sin fråga eller sitt utsökningskrav på sitt eget modersmål och det är upp till systemet att tolka frågan. Notera att det verkligen är frågan om att systemet skall förstå en användares önskemål. System som i hög grad förlitar sig på att känna igen vissa enstaka och nödvändiga ord kan knappast sägas förstå naturligt språk. Den typen av system brukar bete sig lustigt eller olustigt åt om frågan formuleras om eller icke-existerande ord används istället. Det är således av vikt att system som accepterar naturligt språk verkligen bygger upp en intern struktur av en fråga och förmår analysera denna till den grad att systemet kan sägas förstå frågan.

Ett antal system förekommer som påstår sig mer eller mindre förstå naturligt språk. Det går inte att avgöra huruvuda dessa system verkligen förstår naturligt språk enbart genom att studera användarmanualer. För att kunna göra detta krävs antingen direkt tillgång till källkod eller möjlighet att få testa och provköra systemen. Tyvärr existerar system som utger sig för att förstå naturligt språk men som när det kommer till kritan enbart förmår känna igen enstaka ord i en text.

---

[36] (Ti86)
[37] (MS75)
[38] (Se77)
[39] (Fo84)
[40] (Ki84)
[41] (Ka85)

Fördelar med naturligt språk

I ett gränssnitt baserat på naturligt språk kan samtliga funktioner som beskrivits i föregående avsnitt realiseras. Det är samtidigt ett nödvändigt krav att samtliga implementeras för att erhålla ett användbart system. Detta innefattar således syntax, semantik, kontext, ambiguitet och svarsanalys.[42] Vidare är möjligheten för en användare att både kunna uttrycka sig i naturligt språk och i sitt eget modersmål, i stället för ett artificiellt språk, en mycket viktig aspekt. De allra flesta användare kan formulera en fråga på sitt eget modersmål utan att för den skull besitta en djup kunskap om vare sig databaser eller logiska scheman. I denna typ av system kan i princip vem som helst använda sig av systemet på ett rationellt sätt. Notera att det fortfarande enbart är förnuftigt att ställa frågor som ligger inom systemets kompetens. Om systemet har information leverantörer och artiklar är det inte meningsfullt att ställa frågor om skatteplanering. Det man vill uppnå med HSQL, nämligen att låta en så stor grupp som möjligt nyttja ett IS, uppnås med denna typ av gränssnitt.

Nackdelar med naturligt språk

Den allvarligaste nackdelen med ett gränssnitt baserat på naturligt språk är att det blir förhållandevis mycket att skriva för en användare. Frågan måste formuleras och delges systemet på något sätt och i dagsläget är det enbart aktuellt med tangentbord. Eftersom naturligt språk har så pass rikligt med uttrycksmöjligheter ger ofta en fråga upphov till flera tolkningar. Att använda ett system som i tid och otid vill verifiera med en användare vilket alternativ som skall väljas är störande. Detta beteende kan dock mildras genom att använda olika heurismer.

Krav

Denna ansats kräver ingen kringutrustning utöver den vanliga. Däremot blir den minsta möjliga arbetsinsats som krävs i princip densamma som att realisera samtliga funktioner i HSQL eftersom ett naturligt språk knappast blir användbart förutan dessa.

Införande av naturligt språk

Ett flertal system finns som samtliga påstår sig hantera naturligt språk. I bilaga 2 ges en mer detaljerad förteckning över existerande kommersiella system.

Forskningbetonade system: IRUS,[43] KRL, LADDER, LIFER, LUNAR, SHRDLU.[44]

Kommersiella system: Online English,[45] ULG,[46] Language Workbench, Natural Language, Lingua, Q & A, Language Craft, English,[47] USL,[48] NLQ och DIID.[49]

---

[42] (Wa84)
[43] (BMS84)
[44] (Wa84)
[45] (CU83)
[46] (IBM81)
[47] (Be86)
[48] (LOZ85)
[49] (OH86)

Sammanfattning - naturligt språk

Naturligt språk framstår idag som den mest attraktiva typen av gränssnitt. Forskningen på området är alltjämt mycket intensiv och inom en snar framtid kommer säkerligen ambitiösa system finnas att tillgå. I dagsläget finns inget kommersiellt tillgängligt system av den typ som tidigare skisserats. Vidare finns inte något av de existerande system som har något av de nordiska språken som aktuellt språk. En tilltalande tanke är att välja ut det system som verkar kraftfullast och anpassa det till de nordiska språken. Om en nyutveckling skall ske av HSQL framstår naturligt språk som den mest lovande kandidaten.

Semi-naturligt språk

Ett semi-naturligt språk som gränssnitt kan närmast betraktas som en blandning av rent naturligt språk och ett menystyrt dialogsystem. En användare av ett semi-naturligt språk har i varje ögonblick en uppsättning ord att välja mellan. Efter varje nytt inmatat ord eller ordval förändras den mängd ord som är möjlig att nu välja mellan.

Ett exempel kan vara på sin plats. Vi vill ställa frågan om vilka leverantörer som har lägst status 30.

Den första menyn som presenteras har två alternativ: Leverantör och artikel. Vi väljer Leverantör. Därefter presenteras en ny meny men denna innehåller namnen på de kolumner som ingår i tabellen för Leverantörer. Av dessa väljer vi LevNamn. Nu presenteras ännu en meny med de logiska operatorerna: och, eller, icke. Denna visas samtidigt som den tidigare. Vi väljer och. Denna meny försvinner och den förra kvarstår. Nu väljer vi status. Till detta anger vi ett villkor som hämtas ur en annan meny. Nämligen större än eller lika med till vilket vi anger värdet 30.

Vi har nu genom att välja i olika menyer konstruerat en fråga:

Leverantör LevNamn och Status>=30

På detta sätt kan således frågor successivt konstrueras genom att välja ur olika ordförråd. Denna teknik kan givetvis kombineras med grafik och mushantering, även om det går utmärkt att realisera med vanlig fönsterhantering. Med fönsterhantering menas att bildskärmen delas in i logiskt olika delar, sk fönster. I varje fönster kan en meny presenteras. De frågor som konstrueras ser till det yttre ut som eller liknar de som går att konstruera i naturligt språk. Dock finns inte samma frihetsgrad i ordval eftersom det varje ögonblick endast finns en fixt antal ord att välja mellan. Det är för all del fallet även i naturligt språk men där är variationsgraden större eftersom synonymer tillåts och meningsbyggnaden kan variera avsevärt.

Fördelar med semi-naturligt språk

Det finns ingen eller minimal risk för såväl syntaktiska som semantiska fel. I varje ögonblick presenteras endast godkända ord och konstruktionsmöjligheter. Även oerfarna användare bör relativt snabbt och enkelt kunna använda ett liknande system. De konstruerade frågorna är snarlika de i naturligt språk, dvs det förekommer ett minimalt inslag av artificiella ord. Samtliga funktioner i det kunskapsbaserade systemet bör kunna implementeras.

Nackdelar med semi-naturligt språk

Den i stort sett enda allvarliga nackdelen är att den språkliga frihetsgraden minskar. Detta kan reduceras genom att utöka menyernas antal och innehåll, men om menyerna blir alltför omfattande mister de sin slagkraft och blir oöverskådliga. Vissa formella konstruktioner kan inte helt undvikas, tex tester och jämförelser samt användandet av logiska operatorer.

Krav

Beroende på ambitionsnivå blir kraven olika. Om sk dolda menyer med mushantering skall användas måste härför lämplig utrustning ingå, dvs grafiska skärmar med mus. Vid en lägre ambitionsnivå med en mindre sofistikerad fönsterhantering bör systemet kunna användas i befintlig datorutrusting.

Införande av semi-naturligt språksystem

I likhet med grafiska gränssnitt finns endast forskningsbetonade system implementerade. Det system som verkar mest utvecklat idag är RTMS,[50] se bilaga 3. Arbetsinsatsen kan till en del liknas den för grafisk gränssnitt. Dock tillkommer kravet på syntaxanalys som kopplas till en grammatik och ett logiskt schema i den minsta möjliga arbetsinsatsen. Detta pga den dynamiska tolkning som sker och för att systemet i varje ögonblick skall kunna presentera en korrekt uppsättning ord eller kommandon.

Sammanfattning av semi-naturligt språk

Ett semi-naturligt språk som gränssnitt förefaller vid en första anblick som en lämplig kandidat för ett slutligt HSQL. Erfarenheter från praktiken saknas dock. Ett par farhågor är att en användare störs av menyer som dyker upp och försvinner i tid och otid. Om flera omfattande menyer är aktuella är risken uppenbar att användaren drunknar i information. Det kan visa sig att den minskade frihetsgraden i språket eller frågeformuleringar blir ett stort handikapp.

Prolog

Prolog, PROGrammering i LOGik, är ett programmeringsspråk som besitter ett antal kraftfulla mekanismer.[51] Prolog har bl a en naturlig koppling till relationsdatabaser.

I relationsdatabaser representerades data i tabeller.

Leverantör

LevNr	LevNamn	Status	Stad
L1	Svensson	20	Stockholm
L2	Nilsen	10	Köpenhamn
L3	Hansen	30	Köpenhamn
L4	Olsson	20	Stockholm
L5	Mäki	30	Helsinki

Fig. 5.1

(TC85)

[51] (CM81)

I prolog skulle motsvarande tabell kunna anges som ett antal sk faktadeklarationer.

Leverantör(L1,Svensson,20,Stockholm).
Leverantör(L2,Nilsen,10,Köpenhamn).
Leverantör(L3,Hansen,30,Köpenhamn).
Leverantör(L4,Olsson,20,Stockholm).
Leverantör(L5,Mäki,30,Helsinki).

Varje relationselement motsvaras således i prolog av ett faktum. Fakta finns lagrade i prologs egen databas. Tabellnamnet motsvarar något som kallas huvudfunktornamn och varje kolumn motsvaras av ett argument till sammma huvudfunktornamn. Ordningen mellan argumenten har betydelse, dvs om status förekommer som tredje argument i tabellen skall det förekomma som tredje argument i varje faktum.[52]

I ett kunskapsbaserat system måste en inferensmaskin finnas. I prolog finns en inferensmaskin inbyggd. Denna kan användas för att göra utsökningar ur den databas som är prologs egen.

Om vi vill ta reda på nr, status och stad för de leverantörer som heter Nilsen kan vi fråga:

<- Leverantör(nr,Nilsen,status,stad).

Svaret blir:

nr = L2
status = 10
stad = Köpenhamn

<- symbolen används för att ange att det är en fråga som avses. Notera att vi använder variabler som argument. Variablerna har till uppgift att slussa data ur databasen till användaren. Variabler kännetecknas av att de inleds med gemener till skillnad från konstanter som inleds med versaler. I exemplet ovan är nr en variabel och Nilsen en konstant. Tal fungerar i princip som konstanter.

Den princip som prolog använder är följande: En fråga har ställts. Inferensmaskinen använder frågan som en mall eller ett sökmönster och jämför med de fakta som finns i databasen, uppifrån och ner, om det finns något fakta som passar in i mönstret. Om frågan består av flera delfrågor åtskilda av OCH sker sökning för varje delfråga från vänster till höger. När ett faktum har hittats som passar frågemönstret görs en markering i databasen att detta faktum har gett upphov till en lösning. Därefter sker sökning för nästa delfråga. När en lösning har funnits för samtliga delfrågor presenteras lösningen för användaren. Observera att variabler passar vilka fakta som helst, däremot kan enbart samma data passa med varandra.

---

[52] (IN85)

Om vi är intresserade av leverantörer som har status som är högre än 20 kan vi fråga:

<- Leverantör(nr,namn,status,stad) OCH status>20.

nr = L3
namn = Hansen
status = 30
stad = Köpenhamn

nr = L5
namn = Mäki
status = 30
stad = Helsinki

Detta är samma princip som för SQL, där konstruktionen som tidigare konstaterats blir:

SELECT LevNr, LevNamn, Status, Stad FROM Leverantör WHERE Status>20.

I exemplet användes OCH men det går även att använda ELLER samt ICKE. De tester och jämförelser som går att använda i SQL finns även i prolog.

Som synes är prolog synnerligen kraftfullt men ändå förvånansvärt enkelt att använda för detta ändamål.

Vad man önskar i ett prologsystem är en direkt koppling till relationsdatabas. Sådana system har utvecklats med olika lösningsfilosofier. För vidare information se referenserna (JVC84), (CW84), (Za84), (BJ84).

Nedan följer några exempel på hur de frågor som presenterades i avsnitt 2 kan formuleras i prolog med samma databas. Resultaten är exakt de samma som när SQL användes.

Artikel(A1,Mutter,Röd,12,Stockholm).
Artikel(A2,Bult,Grön,17,Köpenhamn).
Artikel(A3,Skruv,Blå,17,Oslo).
Artikel(A4,Skruv,Röd,14,Stockholm).
Artikel(A5,Kam,Blå,12,Köpenhamn).
Artikel(A6,Kugge,Röd,19,Stockholm).

LevArt(L1,A1,300).
LevArt(L1,A2,200).
LevArt(L1,A3,400).
LevArt(L1,A4,200).
LevArt(L1,A5,100).
LevArt(L1,A6,100).
LevArt(L2,A1,300).
LevArt(L2,A2,400).
LevArt(L3,A2,200).
LevArt(L4,A2,200).
LevArt(L4,A4,300).
LevArt(L4,A5,400).

Sökning med testvillkor

Fråga: Leverantörsnummer och status för leverantörer i Köpenhamn

<- Leverantör(levnr,_,status,Köpenhamn).

Notera att _-symbolen används som argument när vi inte är intresserade av vilka värden som är aktuella i det argumentet. Den fungerar i övrigt som en variabel och kallas den anonyma variabeln.

Sökning med eventuella dubbletter

Fråga: Artikelnummer för samtliga artiklar som levereras

<- LevArt(_,artnr,_).

Att generera unika lösningar går bra i prolog genom att använda ett sådant primitiv.

Sökning utan dubbletter

Fråga: Artikelnummer för samtliga artiklar som levereras utan dubbletter

<- UNIKA(LevArt(_,artnr,_)).

Visa all data

Fråga: Samtliga uppgifter om leverantörer

<- Leverantör(levnr,levnamn,status,stad).

Sökning med villkor

Fråga: Nummer för leverantörer i Köpenhamn med status större än 20

<- Leverantör(levnr,_,status,Köpenhamn) OCH staus>20.

Sökning med villkor där resultatet är sorterat

Fråga: Leverantörsnummer och status för leverantörer i Köpenhamn sorterade i sjunkande ordning över status

<- SORTERA(Leverantör(levnr,_,status,Köpenhamn) OCH status>2 status, Sjunkande).

Sökning i flera tabeller

Fråga: För varje artikel, ta fram artikelnummer och städer för motsvarande leverantörer som levererar dessa artiklar

<- UNIK(Artikel(artnr,_,_,_,_) OCH LevArt(levnr,- artnr,_) OCH

Leverantör(levnr,_,_,stad)).

Sökning med nästlade villkorliga sökningar

Fråga: Leverantörsnamn för leverantörer som levererar artikel med nummer A2

<- Leverantör(levnr,levnamn,_,_) OCH LevArt(levnr,- A2,_).

Sökning i nästlade nivåer

Fråga: Ta fram namn på leverantör som levererar minst en röd artikel

<- Leverantör(levnr,levnamn,_,_) OCH LevArt(levnr,- artnr,_) OCH
Artikel(artnr,_,Röd,_,_).

En annan ytterligt kraftfull mekanism hos prolog är möjligheten att definiera regler. En regel består av en villkorsdel och en slutsatsdel.

Grossist(levnamn) OM Leverantör(levnr,levnamn,status,stad) OCH
LevArt(levnr,artnr,antal) OCH antal>=400.

Vi tolkar regeln som att: Ett objekt är en grossist om denne är leverantör och levererar artiklar till ett antal av minst 400 st.

Det som ingår i villkorsdelen kan mycket väl vara andra regler eller fakta som i detta fall. Vi kan nu ställa frågan om det finns några grossister:

<- Grossist(namn).

namn = Svensson
namn = Nilsen
namn = Olsson

En av finesserna med att koppla ihop prolog med ett DBHS är att det skall gå att använda prolog som frågespråk mot en databas. Precis som med SQL krävs att man vet vilka relationer som är aktuella, dvs deras namn och antal argument samt deras betydelse. I ett rent prologsystem erhålls ringa eller ingen hjälp vid felaktiga frågor. Därför skall ett kunskapsbaserat system som finns bakom prolog ge den hjälp som behövs enligt tidigare angivna krav.

Fördelar med Prolog

I jämförelse med SQL är det mycket enkelt att uttrycka komplexa frågor i prolog. En användare kan själv definiera egna regler för just sina behov. Detta motsvarar i princip möjligheten att i SQL definiera egna vyer (eng view). Både syntaxen och semantiken hos prolog är mycket enkel och med ringa träning kan mycket komplexa frågar ställas även av sporadiska användare. Tanken är även här att både syntax och semantik hos en fråga skall analyseras vilket gör prologsystemet mer användarvänligt än vad är fallet idag.

Nackdelar med Prolog

Prologs syntax är trots sin enkelhet inte ett naturligt språk, dvs en viss utbildning och träning i språket är oundviklig. Eftersom prolog till sin natur är baserad på enskilda relationselement snarare än mängder som SQL måste speciella primitiv som hanterar mängder införas, vilket bidrar till att förstöra klarheten hos en fråga ställd i prolog. I övrigt gäller i princip samma invändningar som för att använda SQL som gränssnitt.

Krav

Att använda prolog ställer inte några krav utöver den som ingår i basutrustningen.

Införande av Prolog-baserade system

Ett mycket stort antal prologimplementeringar förekommer varav kan nämnas: PC-Prolog,[53] Quintus-Prolog, Prolog-2.

Ett prologsystem med de funktioner som tidigare presenterats finns ej idag. Det finns heller ingen enhetlig standard för hur kopplingen mellan prolog och en databas konceptuellt skall se ut, vilket medför att, i den mån en koppling till en databas finns, denna är i hög grad varierande från system till system.

Sammanfattning - Prolog

Även om prolog snarare skall betraktas som ett programmeringsspråk än ett datamanipuleringsspråk kan det mycket väl användas som frågespråk mot en databas. De invändningar som finns är de som gäller för att använda SQL som frågespråk, dvs de personer som idag har problem med ett frågespråk kanske inte är hjälpta av prolog trots utbyggnad med ett kunskapsbaserat hjälpsystem.

HSQL - en sammanställning

I den här rapporten har beskrivits ett antal olika tekniker som kan användas för att implementera ett hjälpsystem för SQL. Vidare har diskuterats möjliga gränssnitt. Den gjorda framställningen skall ses som dels en introduktion till området som sådant dels en antydan om vad som bör beaktas vid en mer detaljerad specifikation av ett kunskapsbaserat hjälpsystem för SQL. En specifikation som sedan kan ligga till grund för en direkt implementering av ett HSQL.

---

[53] (IN85)

Referensförteckning

(Am85) A. Amickam

On the Meaning of English Queries to Data Bases, 1985
Department of Computer Science
University of Queensland
St, Lucia, Australia

(Be86) J. Becker

When will computers read Shakespeare
Expert Systems User, Februari 1986, sid 10-15.

(BJ84) M.Brodie, M. Jarke

On Integrating Logic Programming and Databases, 1984
Computer Corporation of America
Four Cambridge Center
Cambridge, Massachusetts 02142

(BL84) J. Bubenko Jr, E. Lindencrona-Ohlin

Konceptuell modellering, Informationsanalys, 1984
SYSLAB, Institutionen för ADB, Stockholms Universitet
106 91 Stockholm

(BMS84) M. Bates, M. Moser, D. Stallard

The IRUS Transportable Natural Language Database Interface, 1984
BBN Laboratories
Cambridge, MA

(CM81) Clocksin, Mellish

Programming in Prolog
Springer Verlag, 1981

(CU83) Cullinet

Online English, 1983
Summary Description
No: SDOE-0383-2002

(CW84) C. Chang, A Walker

PROSQL: A Prolog Programming Interface with SQL/DS, 1984
IBM Research Laboratory
San Jose, Ca 95193

(Da81) C.J. Date

An Introduction to Database Systems.
Addison-Wesley, Third edition, 1981

(Fe81) E. Feigenbaum

Innovation and Symbol Manipulation In Fifth Generation Computer System, 1981
Computer Science Department, Stanford University
Stanford, California 94305

(Fo84) D. Fogg

Lessons from Living In a database. Grafical Query Interface, 1984
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, Mass 02139

(Hi84) J. Hirschberg

Anticipating False Implicatures. Cooperative Responses in Question-Answer System, 1984
Department of Computer and Information Science
Moore School/D2
University of Pennsylvania, Philadelphia PA 19104

(JCV84) M. Jarke, J. Clifford, Y Vassiliou

An Optimizing Front-End to a Relational Query System, 1984
Graduate School of Business Administration
New York University, 90 Trinity Place
New York, NY 10006

(IBM81) ULG, User Language Generator, 1981

Programmer Description, Operation Manual
No: 5788-ADN

(IBM83) DB/2 Introduction to SQL.

IBM File No S370-40, 1983

(IN85) SU-TVT Infologics AB

PC-Prolog, Reference manual, 1985
Box 22
S-182 11 Danderyd, Sverige

(Ka85) H. Kangassalo

A Definitional Conceptual Schema As a Query Interface of the Information System, 1985
Department of Mathematical Science, Computer Science
P.O Box 607
SF-33101 Tampere, Finland

(Ki84) R. King

Sembase: A semantic DBMS University of Colorado
Department of Computer Science
Boulder, Colorado 80309

(LO71) M. Ljung, S. Ohlander

Allmän grammatik
Liber Läromedel, 1971

(LOZ85) H. Lehman, N. Ott, M. Zoeppritz

A Multilingual Interface To Databases, 1985
IBM Germany, Heidelberg Scientific Center

(Lu85) P. Lubonski

Natural Language Interface for a Polish Railway Expert System.
Published in Natural Language Understanding and Logic Programming,
V. Dahl, P. Saint Dizier, North Holland, 1985.

(Ma84) E. Mayes

A Temporal Logic for Reasoning About Changing Data Bases in the Context of Natural Language Questing-Answering, 1984
Mathematical Science Department
IBM T.J Watson Research Center
P.O Box 218
Yorktown Heights, NY 10598

(Mo84) A. Motro

Query Generalisation: A Technique for Handling Query Failure, 1984
Department of Computer Science
University of Southern California, Los Angeles, CA 90089
Technical Report TR-84-308

(Na77) S. Nachmens

Datasystem och datorsystem
Studentlitteratur, 1977

(Ni80) N Nilsson

Principles of Artificial Intelligence
Springer Verlag, 1980

(OH86) K. Obermeier, D. de Hilster

A Natural Language Interface For Database Management Systems, 1986
Battelle, Colombus Laboratories, 505 King Avenue
Columbus, Ohio 43201

(Or83) I. P. Orci

Contributions to Automatic Programming Theory
A Study in Knowledge-Based Computing, 1983
Royal Institute of Technology, University of Stockholm
Department of Information and Computer Science

(Pi83) M. Pilote

A Data Modeling Approach to Simplify the Design of User Interfaces, 1983
Department of Computer Science
University of Toronto
Toronto, Canada

(PW80) F. Pereira, D. Warren

Definite Clause Grammars for Language Analysis.
A Survey of the Formalism and a Comparison with Augmented Transition Network
Artificial Intelligence, Vol 13, s 231-278, 1980

(Re81) P. Reisner

Human Factors Studies of Database Query Languages: A Survey and Assessment, 1981
IBM Research Laboratory
San Jose, California 94193

(Ro84) N. Rowe

Steps Towards Parsing of Query Sequences to a Database, 1984
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943

(Se77) M. Senko

Foral LP - Making Pointed Queries With a Ligth Pen, 1977
IBM Thomas J. Watson Research Center
Yorktown Heights, New York

(So85) J. F. Sowa

Conceptual Structures
Information Processing in Mind and Machines
Addison Wesley, 1985

(Sp84) D. Spooner

Database Support for Interactive Computer Graphics, 1984
Mathematical Science Department
Rensselaer Polytechnic Institute
Troy, NY 12181

(TC85) C. Thompson, S. Corey, M. Rajinikanth, P. Bose, S. Martin, R. Enand, R. Roberts

RTMS: Toward Close Integration between Database and Application, 1985
Computer Science Lab.
Texas Instruments Inc.
Box 226015 M/S 238
Dallas, TX 75266

(Ti86) H. Tiihonen

LINDA project summary, 1986
Technical Research Centre of Finland
Laboratory for Information Processing
Lehtisaarintie 2 A
SF-00340 Helsinki, Finland

(TV72) P af Trampe, Å Viberg

Allmän språkteori och grammmatik
Liber Läromedel, 1972

(Wa84) M. Wallace

Communicating With Databases In Natural Language
Ellis Horwood Limited, 1984

(Wa85) D. A. Waterman

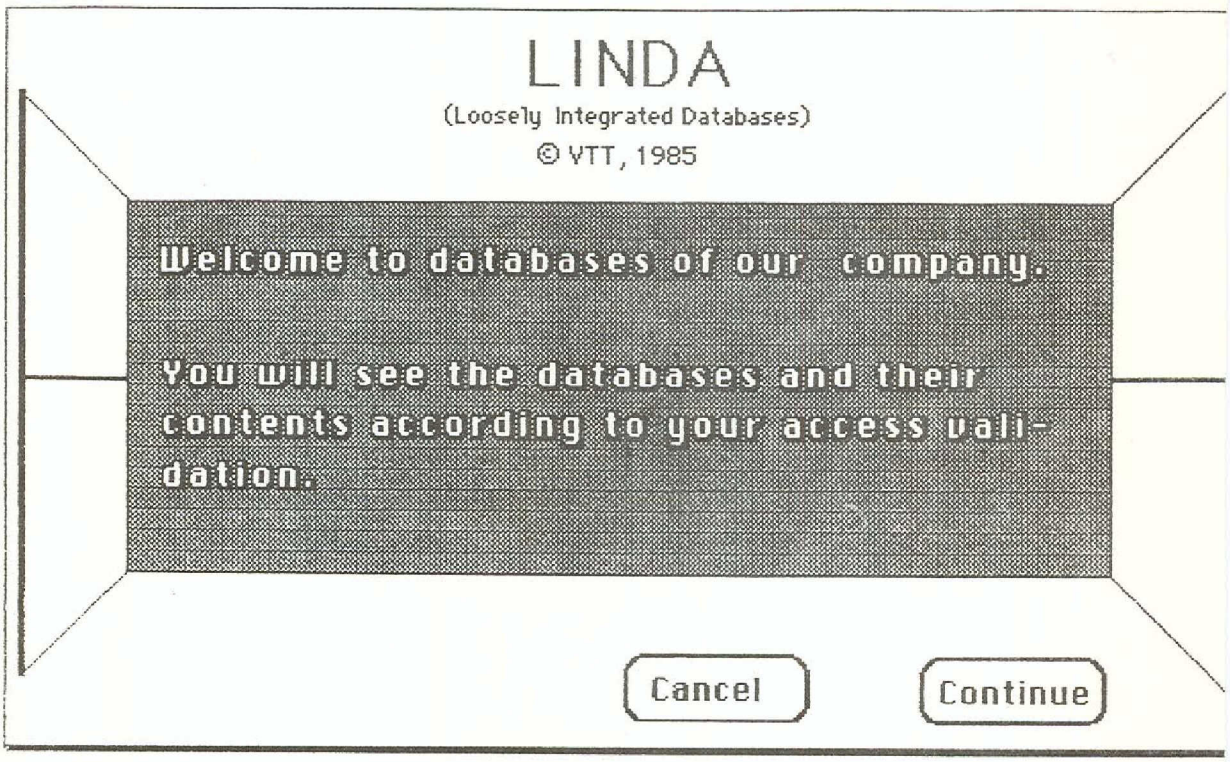
A Guide to Expert Systems
Addison Wesley, 1985

(Za84) C. Zaniolo

Prolog: a Database Query Language for All Seasons
Micro Electronics and Computer Technology Corporation
Austin, Texas

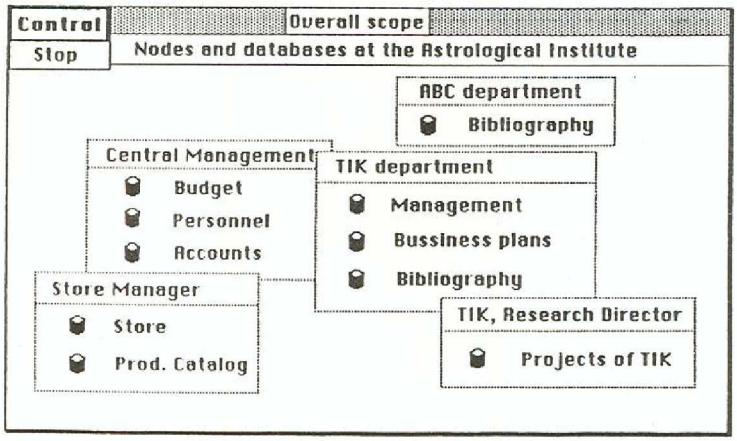
Bilaga 1

Nedan följer ett exempel på hur en session kan se ut med ett grafiskt gränssnitt. Bilderna är hämtade från LINDA. (Se referens (Ti86).)

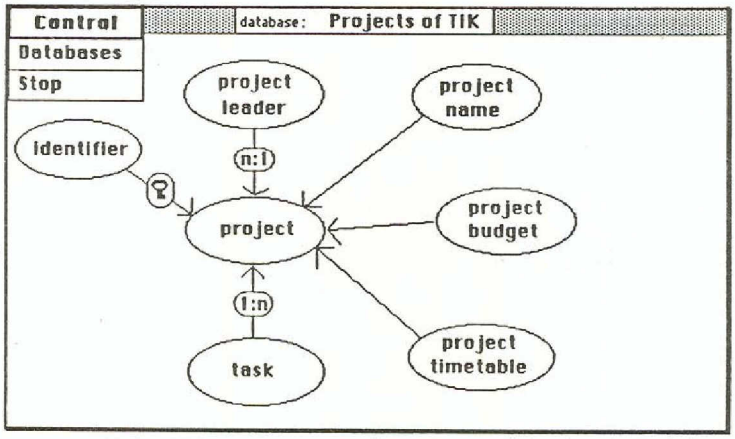


Introduktionsbild

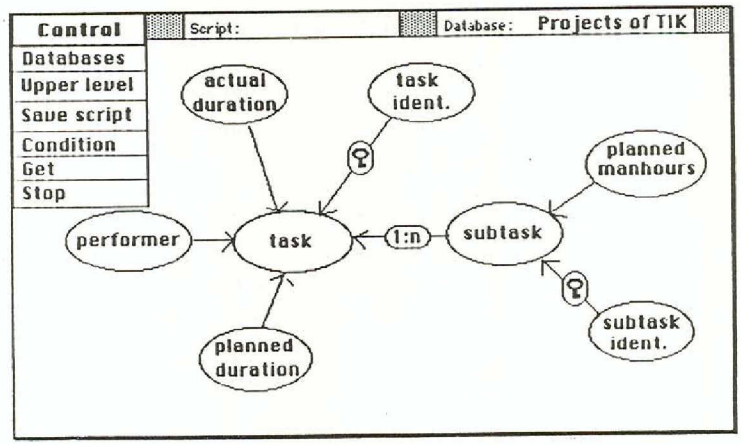
Databaser som kan användas



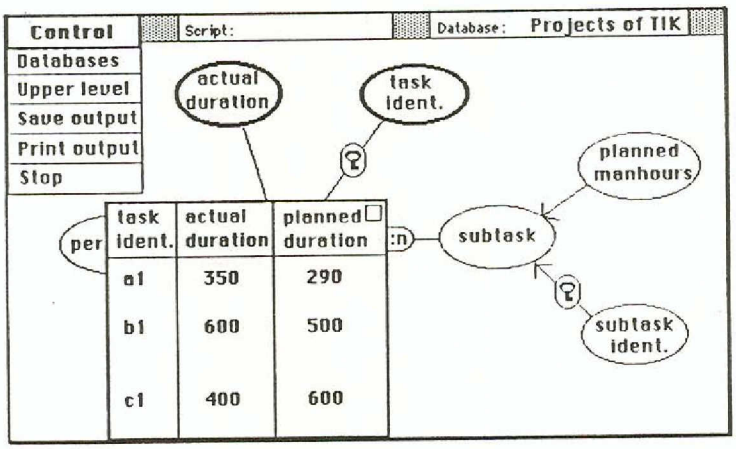
Struktur för databasen "Projects of TIK"



Struktur för objekt "task" i föregående bild



Data för objekt task



Bilaga 2

Förteckning över system med naturligt språk som gränssnitt. Dessa är hämtade från följande referenser: (Be86), (CU83) och (IBM81). För samtliga system gäller att de hanterar engelska. Om något annat är fallet anges detta.

Language Workbench

Detta system är en verktygslåda för programmerare att bygga gränssnitt i naturligt språk. Kan kopplas samman med olika typer av applikationer. Ordförrådet kan anges och utökas av systembyggaren. Tolkaren är en kontext-fri grammatik och hanterar ambiguiteter. Datorer: IBM PC/XT och AT med kompatibler.

Natural Language

Används som gränssnitt till McDonnell Douglas's Reality DBHS. Ordförråd på 300 ord men kan utökas. Endast aktuellt lagringsutrymme sätter övre gräns. Tolkaren bygger på mönstermatchningsteknik. Datorer: McDonnell Douglas samtliga maskiner under Reality som OS.

Lingua

Transformerar engelska eller franska till (första ordningens?) predikat logik att användas i användarapplikationer. Dessa måste vara skrivna i LISP, Prolog, Ada eller Pascal. Ordförrådet kan expanderas. Systemet påstås klara av samtliga meningar i engelska (!). Realiserad med en ATN-tolkare. Datorer: Samtliga persondatorer. Hög grad av portabilitet.

Q & A

Innehåller ett DBHS, ordbehandlare som innefattar tolkare för naturligt språk. Klarar av att uppdatera databasen. Basordförrådet innehåller 400 ord. Kan utökas, endast lagringsutrymmet sätter en övre gräns. Datorer: IBM PC/XT och AT samt kompatibler.

On Line English (Intellect)

Skall kunna hantera både frågor och datamanipulering. Innehåller ca 500 ord initialt men kan utökas utan övre gräns. Tolkning genom analys av både struktur och semantik. Ansett som det första kommersiella systemet för att hantera naturligt språk. Datorer: Samtliga IBM 370, 43xx, 303x, 308x och 309x. Dec-Vax och Terradata DBS 10112.

Language Craft

Är en verktygslåda för att bygga gränssnitt i naturligt språk. Kan blandas med grafik. Strukturanalys av språket genom inprogrammerad kunskap om engelska. Skall vara både robust och tolerant mot syntaxfel och ogrammatiska meningar. Datorer: Dec- Vax under VMS, Symbolics 3600 serien och Explorer.

English

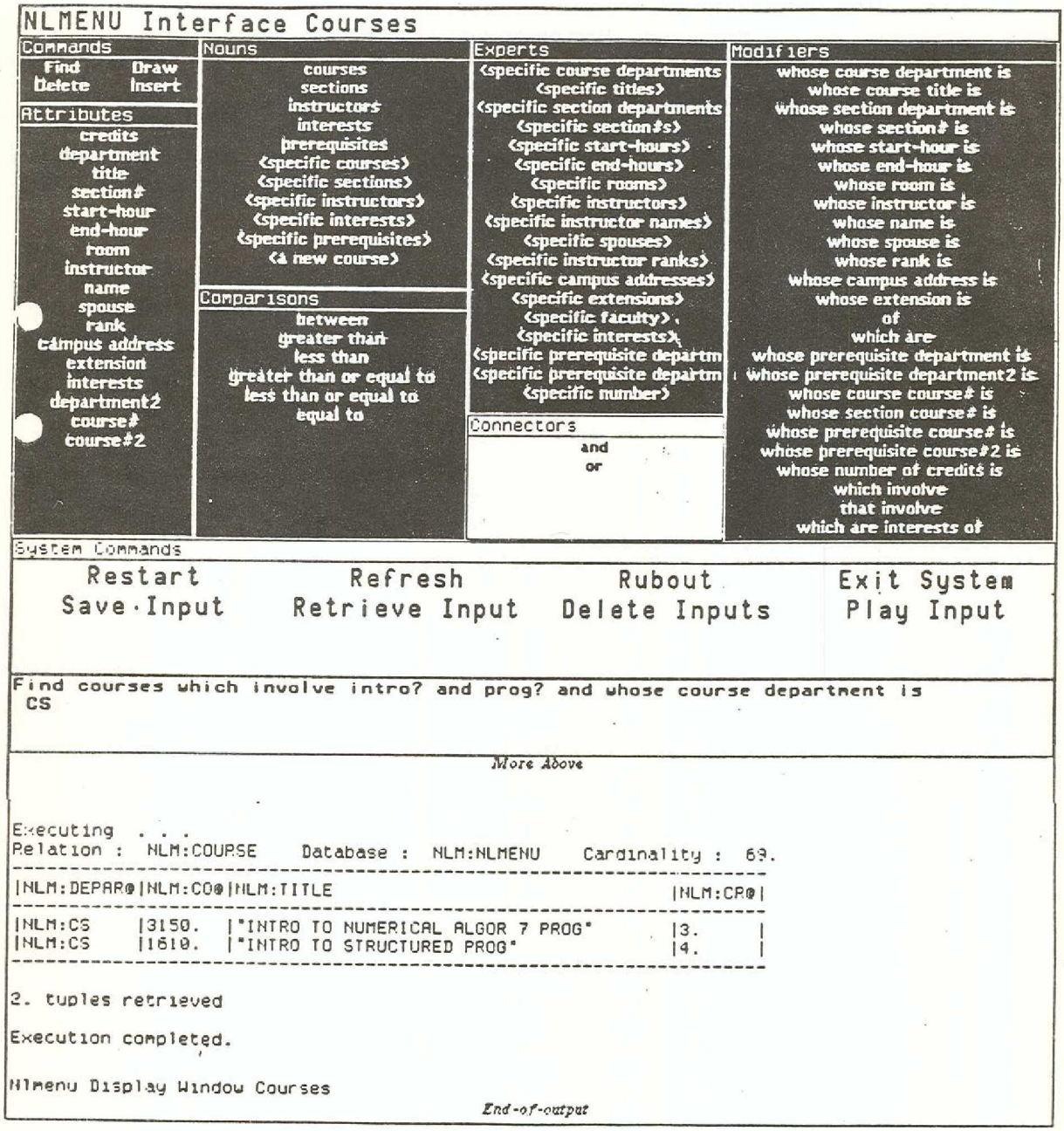
Skall vara i hög grad anpassningsbar till enskilda kunder. Tolkaren skräddarsys för varje applikation. Lämplig som gränssnitt för databaser. Innehåller 300 ord initialt men kan utökas av användaren. Grammatiskt inkorrekta meningar förstås delvis. Datorer: Förekommer på ett antal datorer. Påstås vara synnerligen lämplig för persondatorer.

User Language Generator

System för att definiera grammatikor i valfritt språk. Analys av både syntax och semantik sker. Lämplig som gränssnitt mot databaser. Lanserades av IBM men ingår inte i programsortimentet eller har tonats ner som kommersiell produkt. Datorer: 370 under VM/SP.

Bilaga 3

Nedan ett exempel på hur en skärmbild kan se ut i ett gränssnitt baserat på semi-naturligt språk. För mer information, se referens (TC85).



Vad är SISU?

Svenska Institutet för Systemutveckling (SISU) bedriver forskning om metoder och hjälpmedel vid utveckling av informationssystem. Verksamheten bedrivs tillsammans med intressenter från näringsliv och statliga myndigheter. SISU bildades 1984.

Institutet genomför ett ramprogram på uppdrag av och med stöd av Styrelsen för Teknisk Utveckling (STU) och Intressentföreningen för Svensk Informationssystemutveckling (ISVI), som är en sammanslutning av svenska företag och organisationer inom offentlig förvaltning. Ramprogrammet är treårigt och utarbetas av ISVI.

Vad är syftet med SISU?

Den forskning som SISU bedriver avser att leverera ny teknik, metodik och kunskap till näringsliv och förvaltning. Verksamheten skall utgöra en bro mellan **forskning** och kunskapsutveckling vid våra universitet och högskolor å ena sidan och **tillämpningsfältet** å den andra.

Hur verkar SISU?

Praktiker och forskare vidareutvecklar tillsammans praktiska tillämpningar ur de resultat som framtagits i svensk och internationell forskning.

I projektform eftersträvas samverkan mellan forskare och organisationer med likartade problemställningar och behov. SISU bedriver tre typer av projekt:

1. Prototyputveckling. Konstruktion och realisering av systemutvecklingshjälpmedel och datorstöd till dessa.
2. Metodutveckling. Utveckling av begreppsapparat, notation, språk, användningsriktlinjer samt praktisk tillämpning och utvärdering av metodik eller prototyper.
3. Utredningar inom olika ämnesområden.

Intressenterna medverkar i projekten, exempelvis genom att deras personal ingår i projektet, testar och utvärderar produkter i sin egen organisation eller genom att medverka i styr- och referensgrupper.

Vad gör SISU?

SISU är aktivt inom följande tre områden:

1. *Informationsspridning och förmedling* av informationsteknologi. Detta sker genom utbildning (seminarier, kurser, symposier och konferenser), utgivning av tidskrifterna SISU Informa och SISU Analys, bevakning av nationell och internationell forskning och utveckling samt genom medverkan i standardiseringsarbete.
2. *Intressenter och SISU bedriver tillsammans kollektiv forskning och utveckling* inom områdena
   * Systemutvecklingsmetodik och miljöer
   * Datorstöd för systemutveckling och förvaltning
   * Data- och databasteknologi
   * Människa/maskininteraktion (dialog)

3. *Tillämpningsprojekt.* Ett tillämpningsprojekt orienteras mot aktuella problem hos de intressenter som deltar i projektet. För närvarande prioriteras projekt av följande slag:

   * RAMATIC-tillämpningar och vidareutveckling. RAMATIC är ett datorstöd för specifikation och utformning av informationssystem. RAMATIC kan anpassas till intressenters egen metodik och notation.

   * Intelligenta datakataloger. Projektet inriktas mot fördjupad kunskap om uppbyggnad av datakataloger i syfte att lägga en grund för datautbyte.

   * Samverkan mellan databaser vilka är baserade på olika datamodeller och databasteknologi. Det övergripande syftet är att dels lösa de tekniska problem som är förknippade med att använda databaser med olika uppbyggnad och därigenom skapa möjligheter till enklare utvinning och bättre återanvändning av data som existerar i olika databaser.

   * AVANCE - ett objektorienterat utvecklingssystem. Projektets mål är att utveckla en metod- och verktygsmiljö som kraftigt ökar produktiviteten för systemutveckling, tillförlitligheten och flexibiliteten i den utvecklade system- och programvaran samt möjligheten att effektivt förvalta system.

SISU ANALYS BESTÄLLNING

Svenska Institutet för Systemutveckling

Box 1250, 163 13 Spånga

Sänd mig förkryssade nummer av SISU analys:

☐ ex Nr. 1 : Konceptuell modellering
Pris 75 kr för ISVI-medlem, 285 kr för icke medlem.

☐ ex Nr. 2 : Några aspekter på Kontorsinformations-system.
Pris 75 kr för ISVI-medlem, 285 kr för icke medlem.

☐ ex Nr. 3 : Grafiskt baserade datorstöd för system-beskrivning.
Pris 75 kr för ISVI-medlem, 285 kr för icke medlem.

☐ ex Nr. 4 : ADA-teknologi.
Pris 75 kr för ISVI-medlem, 200 kr för icke medlem.

☐ ex Nr. 5 : Databaser - enkla att hantera (NYTT)
Pris 75 kr för ISVI-medlem, 285 kr för icke medlem.

☐ Sänd mig SISU informa kontinuerligt.

Namn:

Befattning:

Avdelning:

Företag/org.:

Adress:

Postnr/adress:

Telefon: